Project No. 80-I2
(Continuation of Project No. 79-B1)

Cooperator:
University of California
Department of Mathematics
Riverside, California 92521

Project Leaders: Dr. J. K. Oddson/Dr. Sudhir Aggarwal Phone (714) 787-3113

Personnel: Curtis Engle, collaborating with Drs. M. M. Barnes, Tom Baker

Project: Navel Orangeworm Research
Modelling Population Dynamics of the Navel Orangeworm on Almonds

Objectives: To refine and validate further the computer simulation model of the dynamics of navel orangeworm in an almond orchard.

Progress: A preliminary version of a computer simulation model that will simulate the population dynamics of a navel orangeworm infestation as it develops in an almond orchard has been developed. In the next phase of the research it is intended to validate and fine tune this model using precise field and laboratory data.

The computer model follows the development of the n.o.w. population throughout the season, keeping track of the insects as they age through the different stages of their life cycle.

The preliminary version of a computer model has been completed that will simulate the population dynamics of a navel orangeworm infestation as it develops in an almond orchard under the influence of weather and grower control practices.

A complex computer model needs to be properly validated before it can be used in a predictive mode. Thus, although the preliminary model is now operating and looks promising, it would be an error to use it experimentally before at least one season of testing against complete field data. There are also many biological assumptions that are used in the model that need to be confirmed. Once this is done, an informed pest control advisor can use the model in conjunction with his experience to advise on proper pest control strategies.

Plans: (1) To further develop and refine the model by obtaining more accurate data on mummy nut drops and insect life cycle parameters; (2) to conduct validation studies of the model using egg trap data, initial population profile sampling and complete temperature information.

Almond Industry Participation $8,000

PDE

I HAVE COPY

UNIVERSITY OF CALIFORNIA, RIVERSIDE

BERKELEY • DAVIS • IRVINE • LOS ANGELES • RIVERSIDE • SAN DIEGO • SAN FRANCISCO

SANTA BARBARA • SANTA CRUZ

80-I2

DEPARTMENT OF MATHEMATICS

RIVERSIDE, CALIFORNIA 92521

December 31, 1980

Mr. Dale Morrison
Director of Research
Almond Board of California
P.O. Box 15920
Sacramento, CA 95813

Dear Dale,

I am sending two copies of my annual report on Project No. 80-I2 as requested. Included as a separate attachment is a listing of the n.o.w. simulation program, as it comes directly from the computer. I hope that this is satisfactory.

Yours sincerely,

Keith

J. Keith Oddson
Associate Professor

JKO/k

MODELLING POPULATION DYNAMICS OF THE NAVEL ORANGEWORM ON ALMONDS

ANNUAL REPORT

Almond Board of California Project No. 80-I2

December, 1980

J. K. Oddson and S. Aggarwal

Department of Mathematics, University of California
Riverside, CA 92521

TABLE OF CONTENTS

SUMMARY

Developmental work on the navel orangeworm computer simulation model has continued along the following lines:

(1) Model features, including the interactive dialogue presented to the user, have been refined and expanded;

(2) Biological parameters and assumptions continue to be updated as new data becomes available from laboratory experiments of C. E. Engle and Dr. M. M. Barnes, Department of Entomology, UCR;

(3) Field validation studies have begun. Initial tests of the program in two real field situations have shown that it successfully predicts the period, but not yet the shape, of the spring oviposition curve. Some model adjustments are now under study and a more extensive validation effort is planned for next season;

(4) Other modifications are being made to the programming syntax, to create a version of the model that can be run on the larger PRIME 400 computer at UCR. This version will be compatible with the statewide IPM computer network and could be accessed by users through terminals at the County level when ready, and that network is in place;

(5) Documentation of the model is in preparation. This will provide a complete technical description of the operation of the program at both internal and user levels, however, a simpler version could also serve as a manual for users.

Model Developmental Work and Initial Validation Studies

Development of a computer model to simulate the population dynamics of a navel orangeworm infestation as it evolves in an almond orchard under the influence of weather and grower control practices was begun last year in the expectation that a better quantitative understanding of the life cycle and crop interactions of this pest should lead to improved measures for its control. By December, 1979 the basic programming structures for a preliminary version of the model had been completed. Since then the project has continued, with developmental work during the 1980 season concentrating on verification (debugging) of existing routines, program modifications and additions, and an initial attempt at field validation.

(a) Model Refinements

Accurate information on the biological parameters of the life cycle of the navel orangeworm was not available for the initial version of the model. Estimates for the temperature thresholds (59°F) and the degree-day development periods of the various stages had to be inferred from literature data (Wade, Hilgardia (1961); Goodwin and Madsen, Hilgardia (1964)) and were inserted provisionally, pending verification. Subsequent work by C. E. Engle and Dr. M. M. Barnes (Almond Board Report, December, 1979) showed that these first threshold estimates were too high. In fact, laboratory experiments placed the temperature threshold for egg hatching at 54.8°F and revised the corresponding thermal development constant.

Preliminary data from similar experiments with larvae indicated that the larval threshold would also be close to 55°F. Orchard observations of nocturnal behavior of moths showed that flight, mating, and oviposition activities continued even at temperatures as low as 56°F, with summer oviposition flights peaking at 10:30 p.m. and mating occurring in another peak just before dawn. This year, on the basis of these experiments, the observed egg developmental parameters and more accurate ovipositional timing have been incorporated into the model and the thresholds for the remaining stages have been revised downward to 55°F. Additional laboratory experiments by Engle and Barnes have been in progress to determine precise values of the biological parameters for these other stages. They will be inserted into the model as they become available. Preliminary indications from these experiments are that the pupal parameters will be close to those already in use but that the adult ovipositional threshold could be as low as 53°F. Initial attempts at model validation (see below) indicate that more detailed experimental information may still be required on inter-instar larval development and adult ovipositional mechanisms and behavior.

Other developmental work has concerned modifications and additions to the input/output routines of the program in preparation for field testing of the basic phenological structure of the model - the heat-summation driving of the population cycle. To increase flexibility for such model validation, the temperature threshold parameters have been defined as variables, to be chosen by the user,

and there are several options for the display of the results of simulation. Numerical and graphical output of day-degree accumulation and predicted populations of adults, eggs and larvae can be followed, daily if desired, as the season moves along. An alternative output procedure returns a count of eggs that would have been collected from a simulated placement of egg traps in the orchard. Comparison of this graph with the actual egg trap counts can be used as a test of the validity of the model.

Output options and the interactive (screen) dialogue by which they are presented to the user are shown in Appendix I. A complete listing of NOW-2, the current version of the Navel Orangeworm Simulation program, is given in Appendix II.

(b) Validation Studies

Validation (field testing) is a crucial (and often lengthy) phase in the development of any simulation model. A program such as NOW-2 cannot be implemented as a useful and reliable tool until it has successfully demonstrated a predictive capability on data from actual field situations. For navel orangeworm or almonds two sources are commonly used to provide pre-season information on the resident overwintering population:

(1) Orchard mummies are sampled for the presence of larvae and pupae. Larval instars can be sorted by head capsule sizing and pupae can be classified as either "early" or "late". Pupal cases may also be encountered - evidence of prior adult emergence. Such an analysis of the infestation data in detail is usually limited by time and other constraints, but it would be useful to have for model validation. In fact, a sequence of such samples, begun early in the year (Jan.-Feb.) before significant emergence occurs (or eggs have reappeared), would provide an accurate biofix on the size and age distribution of the insect population while it is still small and closely grouped within the larval and the pupal stages. This information could be used to initialize the model and to provide some control checks as the simulation proceeds.

(2) Ovipositional traps are emplaced (weekly) during April-May to bracket spring n.o.w. emergence for the timing of May sprays. Since they provide data on only a single stage of the insect life cycle, they give less direct information on the size and structure of the whole population than can be obtained from detailed mummy sampling. Moreover, there seem to be no significant correlations among infestation levels and egg-trapping results, at least with bait material which is presently in use (Engle and Barnes, Almond Board Report, 1980). Nonetheless, these data are relatively easy to obtain and they can be used to test the ability of the model to predict at least the timing and the shape of the spring ovipositional curve. If there is little overlap between emerging and F1 generation adults, it should also be possible to use these data directly as initial input to the simulation model.

Two validation trials are reported here. In each case the model was first initialized with a pre-season profile of the navel orangeworm population, as determined from winter sampling of mummified nuts, and then simulated forward on the basis of degree-day heat unit summations through the spring months and asked to predict the occurrence of spring oviposition. The results were compared with actual field egg-trapping data, to provide the first tests of model validity. Field data used in the trials were collected by C. E. Engle during the 1979 and 1980 seasons from portions of a Superior Farms orchard in Kern County and have been reported elsewhere (Engle and Barnes, Almond Board Reports, 1979 and 1980).

Table 1: Mummy Sampling Profiles of the Overwintering N. C.W. Population in Superior Farms Orchard Plot R94C (C. E. Engle, 1979).

1979 Sample Data	Insect Counts (Per 100 Nuts) Larval Instars 1	2	3	4	5	6	Pupae	Total
1/26	-	2.0	22.2	62.7	46.5	42.4	14.1	189.9
3/2	-	-	22.6	40.8	29.5	53.8	23.4	170.1
3/22	-	-	5.9	25.1	32.4	45.7	36.9	146.0
3/30	-	-	-	13.6	31.7	46.8	52.9	145.0
4/6	-	-	-	7.7	16.6	38.4	65.3	128.0
4/14	-	-	-	3.5	10.4	29.9	70.2	114.0

Table 2: Mummy Sampling Profiles of the Overwintering N. C.W. Population in Superior Farms Orchard Plot R94X (C. E. Engle, 1980).

1980 Sample Data	Insect Counts (Per 100 Nuts) Larval Instars 1	2	3	4	5	6	Pupae Early	Late	Total
2/9	-	-	5.0	29.0	47.0	62.0	15.0	10.0	168.0
3/17	-	-	-	5.0	14.0	13.0	25.0	22.0	79.0
4/10	-	-	-	3.1	3.5	7.0	17.1	21.1	51.8
4/17	-	-	-	1.6	5.9	9.1	21.7	28.4	66.7
4/24	-	-	-	1.6	3.9	2.7	9.3	21.8	39.3

In Tables 1 and 2 are shown the larval-instar/pupa censuses of the over-wintering n.o.w. population and its progression through time, as determined from sequences of mummy samplings taken during the winter months of 1979 and 1980 and adjusted to a standard sample size of 100 nuts. No adults were observed during these periods, however they must have been present in the orchard for in both years eggs began to appear on the sampled nuts by early April. Even though these moths were not directly observed, their presence can still be inferred indirectly from the tables of data. If successive rows are subtracted, from top to bottom, and then columns added cummulatively from left to right, the result is a new table of values which represent the number of insects (per 100 nuts) that must have passed out of one category and into the next during the time periods between successive samplings. In particular, the final column reveals the adult emergence which must have occurred.

Table 3 shows the results of these calculations as applied to the 1979 data of Table 1. A steady maturation of the population is evident, with adult emergence occurring throughout the winter and spring months. Those adults which emerge very early in the year usually may be ignored, for they will die before night temperatures have risen enough to permit mating and oviposition, but those emerging later (e.g. March) could survive to reproduce. For this reason the simulation for 1979 was begun as early as possible, using the sample data of 1/26, in order to avoid the complicating factors of the presence of viable moths.

Table 3: Progression of Overwintering N.O.W. in Superior Farms Orchard Plot R94C (1979).

Insect Numbers (Per 100 Nuts) Passing From						During Period
Instar 2 to 3	Instar 3 to 4	Instar 4 to 5	Instar 5 to 6	Instar 6 to Pupa	Pupa to Adult	
2.0	1.6	23.5	40.5	29.1	19.8	1/26 - 3/2
0	16.7	32.4	29.5	37.6	24.1	3/2 - 3/22
0	5.9	17.4	18.1	17.0	1.0	3/22 - 3/30
0	0	5.9	21.0	29.4	17.0	3/30 - 4/6
0	0	4.2	10.4	18.9	14.0	4/6 - 4/14

A similar analysis of the 1980 data of Table 2 is shown in Table 4. The negative values in the 3rd row indicate some problem with either the 4/10/80 or the 4/17/80 sample - these data are not consistent with a maturing population progression. Moreover, since fewer samples had been taken during the winter months of this year, it was more difficult to obtain an accurate population profile before emerging adults had become a significant factor. To take this into account the simulation for 1980 was initialized with the population profile of the 3/17 sample together with 89.0 adult moths which, according to Table 4, had emerged between 2/9 and 3/17. The moths were inserted (rather arbitrarily) in 3 separate cohorts, equally spaced along the life span.

Table 4: Progression of Overwintering N.O.W. in Superior Farms Orchard Plot R94X (1980).

Insect Numbers (Per 100 Nuts) Passing From						During Period
Instar 3 to 4	Instar 4 to 5	Instar 5 to 6	Instar 6 to Early Pupa	Early Pupa to Late	Late Pupa to Adult	
5.0	29.0	62.0	111.0	101.0	89.0	2/9 - 3/17
0	1.9	12.4	18.4	26.3	27.2	3/17 - 4/10
0	1.5	-.9	-3.0	-7.6	-14.9	4/10 - 4/17
0	0	2.0	8.9	20.8	27.4	4/17 - 4/24

Results from the two simulations are shown in Figures 1 and 2, where predicted patterns of spring oviposition are compared with actual values of field egg-trapping counts. (Peak vertical dimensions have been scaled to be comparable - only relative heights and their positions with time are relevant for these preliminary tests of the phenological model.) In each case the spring ovipositional period has been predicted quite well, however there are discrepancies in the distribution of the peaks of the curves which indicate that the model is not yet simulating the behavior of all stages of the insect life cycle correctly. Preliminary analysis suggests that two factors are primarily responsible:

(1) The large narrow peak which appears near April 17 in the simulation trial for 1980 was produced by a group of mature adults which emerged some time earlier but had been prevented from laying eggs previously because evening

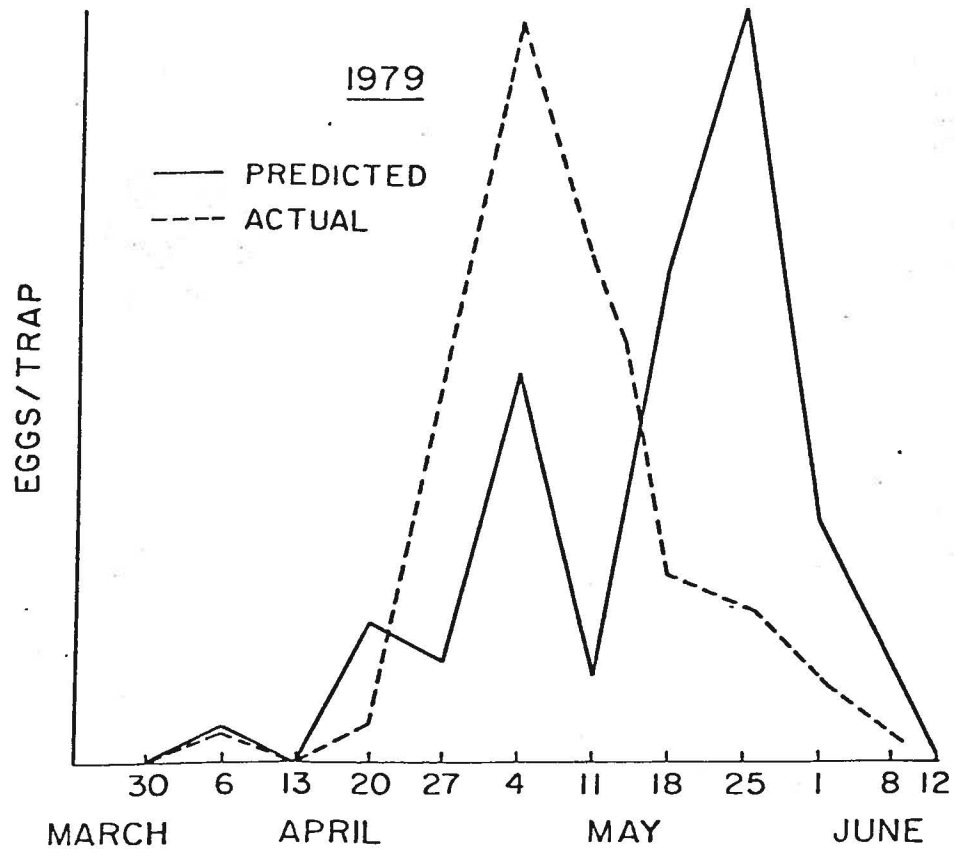

Figure 1. Simulated and Actual Ovipositional Trap Data in Superior Farms Orchard Plot R94C

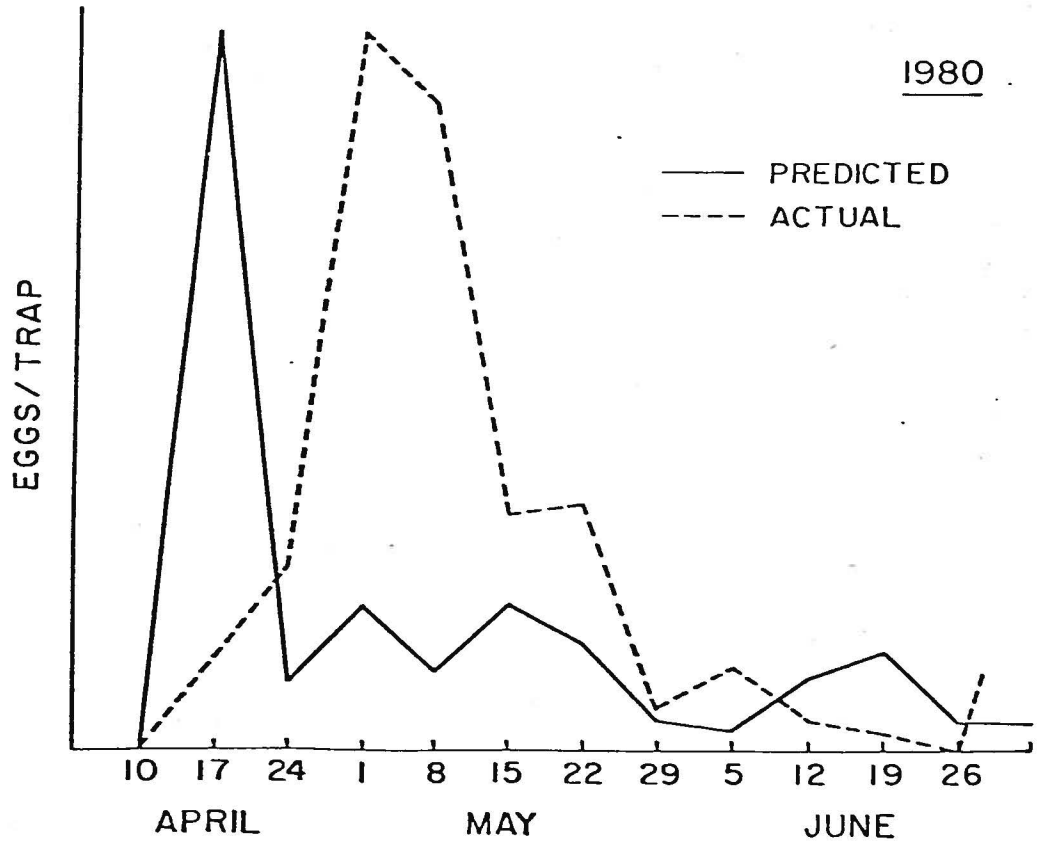

Figure 2. Simulated and Actual Ovipositional Trap Data in Superior Farms Orchard Plot R94X

temperatures had been too low for mating/ovipositional flights to occur. As soon as temperatures rose above the threshold value prescribed in the model the deferred oviposition occurred in a burst. This factor was not as important or evident in the 1979 simulation, for during the corresponding spring period the weather was warmer and a low temperature cut-off was seldom invoked. The system is therefore quite sensitive to the biological assumptions which may be applied concerning ovipositional mechanisms and behavior, especially in early spring, before evening temperatures have risen much above thresholds. Very little is known about this behavior. Some oviposition experiments have been conducted by Engle and Barnes (Almond Board Report, 1980) but more laboratory work may still be required in order to form a satisfactory experimental basis for model adjustment.

(2) The simulation graphs are less regular, with more internal peaks, than the field data curves. This can be accounted for by the current manner in which the initial population profile is put into the model. Larvae, sorted into 6 instars, are always placed at the beginning of their instar stage and it is assumed, lacking experimental evidence to the contrary, that one-sixth of larval development occurs in each instar. No allowance has been made for inserting any finer, intra-instar age structure, which of course must be present in practice. As a result the overwintering larvae of the initial population profile pass through their life cycle in groups widely separated in age, leading to the discrete, multiple oviposition peaks seen in the simulation graphs of Figures 1 and 2. A more flexible input procedure which will permit the entry of an initial population profile with any age structure continuum is now being prepared. Laboratory experiments to obtain more information on inter-instar larval development are to be arranged with Engle and Barnes.

Validation trials and model adjustments will continue next year. The intention is to conduct a series of detailed early winter orchard samplings of mummified nuts in the hope that an analysis similar to that in Tables 3 and 4, but including the effects of heat unit ageing, can provide an accurate fix on the fine structure of the age distribution of the overwintering population. Of course this would not be a practical approach in field applications. It is to be used, together with egg-trapping data and the new input procedure, only to help achieve proper validation of the model. The ultimate goal of these studies continues to be the development of a predictive tool that will be reliable, and yet simple enough for practical field implementations.

IPM Network Transfer Compatibility

Although the n.o.w. simulation model is being developed on the TERAK microcomputer, a second aspect of the work concerns the preparation of the model for eventual transfer to the statewide IPM computer network, where it would be accessible to farm advisors and researchers through county offices and other locations in the state. The size and speed of this system will facilitate the handling of large scale simulations. The computing equipment for the network has been purchased from Prime, Inc., and will be put in place during the next few months. It consists of a central Prime 550,to be housed on the campus at U.C. Davis and tied into three Prime 250 district computers at Davis, Riverside and Parlier. Dedicated telecommunications lines will link the district computers to terminals in county offices and elsewhere.

The transfer of a computer program from one computer to another involves problems of compatibility of the operating systems of the machines and the programming languages they support. The n.o.w. simulation program is written in the UCSD version of Pascal. Prime has its own version of "Standard" Pascal, which will be available on the IPM network. While these two versions are somewhat different, the modification of a program from one to the other should have been a routine, although technical, matter. Moreover, UC Riverside already has a Prime 400 computer and was selected by Prime, Inc. as a test site for their Pascal system prior to marketing. This seemed to provide a unique opportunity to prepare the n.o.w. simulation model for transfer to the statewide IPM network at the same time that it was being developed on the microcomputers. Unfortunately the Prime Pascal system was itself under development and had a number of problems, including the following:

(1) File handling

(a) All temperature inputs used in the n.o.w. program must be inserted in a "File of reals", which is an array of data organized in a manner understandable to the computer. To create this file a small program must be written to convert the actual temperatures and rewrite them to a datafile of reals. In this process the file of temperature data must be opened for reading and an output file of reals created for writing. The Prime Pascal system would open the temperature file but then give an error message when it tried to read it, complaining that the file had not yet been opened. This was corrected after 1 1/2 months, with a revision of the Prime Pascal compiler.

(b) After installation of the new compiler the computer would read the temperature file all right, but when all the data had been read from the file it would stop and do nothing, perhaps caught in some type of an infinite loop. This problem was corrected by Prime, Inc. in about another week.

(2) Strings

"String variables" are permitted in the UCSD version of Pascal but not in "Standard" versions. Conversions were made by using an "array of characters", but then in order to implement "string handling" a whole new set of array handling routines had to be developed. This problem coincided with Prime's file handling problems so no additional time was lost in waiting for these functions to be developed.

(3) Missing data

The reserved Pascal word NEW refers to a function that will allocate storage space for records that are needed during program execution. It is used extensively in the n.o.w. program but was left out of the Prime Pascal function library. As a result the program would compile but could not be run. The function NEW was finally added to the Pascal library early in October, when a new operating system was installed in the Prime 400.

Most of the serious difficulties now seem to be resolved. The remaining syntax modifications should be completed next winter, leading to a version of the model that can be run on the Prime 400 and, after validation, on the IPM network.

On a related matter, documentation which will fully explain all program routines is now being prepared.

Appendix I: Output Options With Computer Screen Dialogue

{Explanatory comments, which do not appear on the screen, are included here in braces. Initially the following main command level prompt line appears on the screen.}

Command Mode: H(elp N(ewdata O(utput S(imulate R(eset Q(uit

If you need help, type "H" when the Command Mode prompt-line appears.

{Type O (for Output)}

Output:

Singlestep (y/n) ?

{Type 'y' (for yes) or 'n' (for no). If 'y', the computer will process the simulated life cycle events one step at a time, stopping at the end of each step. This is useful for debugging purposes, to follow the model one step at a time. If 'n', simulation will proceed continuously but it can always be interrupted by pressing the console 'space bar' and restarted by typing 'S'.}

Do you want basic output after each event?

{Type 'y' (for yes) or 'n' (for no). If 'y', then after processing each simulated life cycle event a readout of updated numerical information on current model time, degree-day accumulation, insect population profile, etc. will be displayed on the screen.}

Do you want a dump of the list after each event?

{Type 'y' or 'n'. This is a feature which is of interest only for debugging the program.}

Do you want egg-trapping data?

{Type 'y' or 'n'. If 'y', then the program will produce as output a count of eggs that would have been collected from a simulated placement of egg traps in the orchard. The following dialogue would appear, to receive information on the egg-trapping dates.}

How many samples taken?

{Type in the desired number (between 1 and 15) of ovipositional trap placement dates to be used.}

Time of first emplacement (Julian day) ?

{Type in the placement date (between 1 and 365)}

Weekly sampling?

{Type 'y' or 'n'. Most egg-trapping samples are retrieved on a weekly basis. If 'y', the program will automatically calculate the date of each retrieval, based upon the previously inserted date of first emplacement and number of placements. If 'n' then these retrieval dates must be specified by the user and the following dialogue will appear.}

Time of retrieval #1, emplacement #2?

Time of retrieval #2, emplacement #3?

.
.

Time of last retrieval?

{In each case the desired Julian calendar date must be specified.}

Printed to R(emote P(rinter?

{The user must specify where the numerical output data is to be sent. If P (for Printer) is indicated, the program will print the data on a time scale from which graphs such as Figures 1 and 2 can be drawn.}

{If the response to the question "Do you want egg-trapping data?" was 'n' instead of 'y', then egg-trapping output will not be simulated. The following dialogue will appear instead.}

Hardcopy?

{Type 'y' or 'n'. If 'y', the program will produce simulation graphs of the numbers of insects in various insect stages as the season progresses. Which stages are to be graphed are determined by the following questions.}

Do you want egg population on graph?

Do you want larval stage #1 on graph?

Do you want the total larval population on the graph?

Do you want adult population on the graph?

{Respond with 'y' or 'n' to each question.}

The present scale factors are

0.05 for eggs
0.50 for larval stage #1
0.02 for total larval population
0.50 for adult population

Change scale factors (y/n)?

{This option is merely to ensure that the graphs will fit properly on the printer paper. If 'y', the following dialogue appears.}

Change scale factor of eggs?

{If 'y', the next question is}

Enter new scale factor →

{Type in the revised scale factor. The dialogue then continues in the following manner}

Change scale factor for larval stage #1?

Change scale factor for total larval population?

Change scale factor for adults?

{In each case 'y' or 'n' is expected. If 'y' in any case, a new scale factor is requested from the user.}

Printed to: R(emote or P(rinter?

{Again the user has the option of sending the output to be printed directly on a time scale graph or to some other output device. When the user responds, the program then exits the Output procedure and returns to the main command level.}

Command Mode: H(elp N(ewdata O(utput S(imulate R(eset Q(uit

December 8, 1980

Almond Board Project No. 80-I2: Modelling Population Dynamics
Project Leaders: J. K. Oddson and S. Aggarwal, Department of Mathematics, UCR

Project Status: Developmental work on the navel orangeworm computer simulation model has continued along the following lines:

(1) Model features, including the interactive dialogue presented to the user, have been refined and expanded;

(2) Biological parameters and assumptions continue to be updated as new data becomes available from laboratory experiments of C.E. Engle and Dr. M.M. Barnes, Department of Entomology, UCR;

(3) Field validation studies have begun. Initial tests of the program in real field situations show that it successfully predicts the period of spring oviposition but does not do as well on the pattern. (See Figure 1). Some model adjustments are now being tested and a more extensive validation effort is planned for next spring;

(4) Other modifications are being made to the programming, to create a version of the model that can be run on the larger PRIME 400 computer at UCR. This version will be compatible with the statewide IPM computer network and could be accessed by users through terminals at the County level when that network is in place;

(5) Documentation of the model is in preparation. This will be a complete technical description of the operation of the program at both internal and user levels, however a simpler version could also serve as a manual for users.

Validation Studies: Data collected by C. E. Engle from portions of a Kern Co., Superior Farms almond orchard during the 1979 and 1980 seasons were used for initial tests of the predictive ability of the model. Simulations were initialized with a pre-season profile of the navel orangeworm population, obtained from winter sampling of mummified nuts, and run through the period of spring oviposition. Ovipositional trap data were used for the check. Figure 1 shows how the simulated and field values compared. The ovipositional periods have been predicted quite well but the distribution of peaks in these curves do not match. Two factors seem primarily responsible: (1) The sytem is quite sensitive to the biological assumptions that have been made concerning ovipositional mechanisms and thresholds, especially during the early spring when orchard temperatures are low; (2) The present method of inserting the initial population profile is not sufficiently accurate - a finer specification of the age structure is required.

Validation trials of this same type are expected to continue next spring, with the addition of (i) laboratory experiments to provide more information on ovipositional behaviour and inter-instar larval development. Some of these are already are in progress; (ii) a sequence of early (January/February) orchard mummy samplings to more accurately determine the age structure of the overwintering population; (iii) modification of program input procedures to permit entry of such age structure detail in the profile of the initial population.

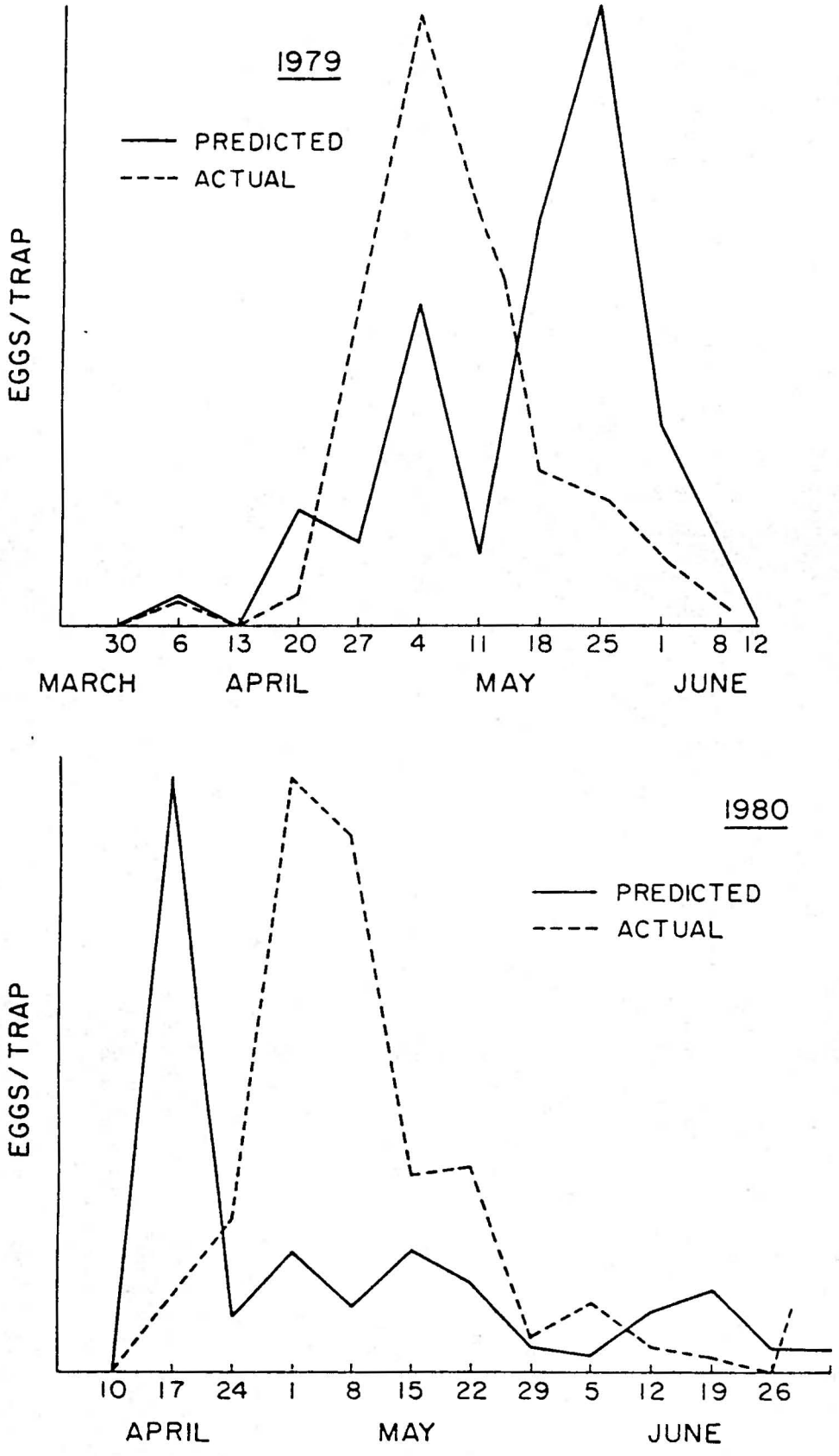

Figure 1. Simulated and Actual Ovipositional Trap Data in Superior Farms Almond Orchard.

Appendix II: Navel Orangeworm Computer Simulation Program, NOW-2

PARAMYELOIS

TRANSITELLA

J. K. Oddson S. Aggarwal
M. Campbell D. Buscaino H. Ford

UCR, 1980

```
PROGRAM DRIVER;

{NOW Program file requiring FOUR include files:   FWD.TEXT ,

                                                    MAIN ,

                                                    NEWDATA.TEXT ,

                                                &   ONE.TEXT        }


{Simulation of Navel Orangeworm infesting an almond orchard: (version 0)
 Includes Sep.24-80 and Sep.19-80 revisions of Sep.17-80 BACKUP disc.
 Source language:  UCSD PASCAL Version I.5  (TERAK)  }


{$S+}


{$I#5:MAIN}    { MAIN contains all constant and variable declarations}

{Segment-able procedures: }

{$I#5:NEWDATA}  {NEWDATA has been made into an include file.}


SEGMENT PROCEDURE INIT; {Initializes global variables}


BEGIN
  WRITELN('Please wait while I perform some initializations');

 {Init's: }
  CSHOVERP:=CSHTREES / POLTREES ;

  DDINLARVA:=DDIN1+DDIN2+DDIN3+DDIN4+DDIN5+DDIN6;

  MINREMIN[6]:=0.0;   {Update uses 1..6 as MINimum d-d's REMaining (to be ac-  }
  MINREMIN[5]:=DDIN6; { cumulated) by a larva IN []'th. instar }
  MINREMIN[4]:=DDIN5 + DDIN6;
  MINREMIN[3]:=DDIN4 + MINREMIN[4];
  MINREMIN[2]:=DDIN3 + MINREMIN[3];
  MINREMIN[1]:=DDIN2 + MINREMIN[2];
  MINREMIN[0]:=DDIN1 + MINREMIN[1]; {Schedule uses 0..5 as:
                                        "MAXREMIN[i]" = MINREMIN[i-1] }

  MINPOP:=0.1;        { Mininum population needed for an event to occur }
  EGGTHR:=55.0;       { Develomental thresholds for eggs, larvae, pupae , }
  LARVATHR:=55.0;     { adults, along with calling/flight threshold. }
  PUPATHR:=55.0;
  ADULTTHR:=55.0;
  CALLTHR:=55.0;
```

```
  {DAYEGGS:=FALSE;}

  SFEGGS:=0.05;       {  Graphing scale factors for eggs,larval stage #1,total  }
  SFLARVA1:=0.5;     {  larvae and adults   }
  SFTOTLARVA:=0.02;
  SFADULTS:=0.5;

  MARK(HEAP);

  NODEFAULT:=NOT ( SENSEFILE(DEFAULT) ) ;
  IF  NODEFAULT
    THEN BEGIN
           WRITELN('The default file, ',DEFAULT,', is not available.');
           {$I-}
           REPEAT
              WRITE('Enter temperature file name--> ');
              READLN(TEMPNAME);
              WRITELN;
              RESET(TEMPS,CONCAT(TEMPNAME,'.TEMP') )
           UNTIL (IORESULT = 0);
           {$I+}
           READFILE;
           CLOSE(TEMPS)
         END
    ELSE BEGIN
           TEMPNAME:=DEFAULT;
           RESET(TEMPS,CONCAT(TEMPNAME,'.TEMP') );
           READFILE;
           CLOSE(TEMPS)
         END
END; {of Init}


SEGMENT PROCEDURE RESTART;
VAR  I:INTEGER;
     K:EVENTKIND;
     T:EVENTPTR;

BEGIN
 {Init/Reset's: }
  STARTIME:=DEFSTART;
  HARVESTIME:=DEFHARVEST;
  STOPTIME:=MAXTIME;
  OLDTIME:=STARTIME;
  CURRENTIME:=STARTIME;

  TOTALDDS:=0.0;

  HASBEGUN:=FALSE;

  {DAYEGGS:=FALSE;}

  FIRSTRAP:=TRUE;
  EGGTRAPCOUNT:=0;
  EGGOUT:=FALSE;



  SINGLESTEP:=FALSE;
  BASIC:=TRUE;
  DUMP:=FALSE;
  HARDCOPY:=FALSE;

  NEXTOUT:=1.0;
```

```
  FREE:=NIL;
  TEMPFREE:=NIL;


  FOR K:=HATCH TO SWEEP DO COUNT[K]:=0;

  FOR I:=-1 TO 8 DO POP[I]:=0.0 ;
  LASTUPDATE:=STARTIME;

  {Now set up Harvest event: }
  NEW(CURRENTEVENT);
  WITH CURRENTEVENT^ DO
    BEGIN KIND:=KNOCK;
          TIME:=HARVESTIME;
          POPULATION:=0.0;
          DEVPATH:=NONE
    END;
  COUNT[KNOCK]:=1;

  {Now set up Final event: }
  NEW(CURRENTEVENT^.LINK);
  WITH CURRENTEVENT^.LINK^ DO
    BEGIN KIND:=FINAL;
          TIME:=MAXTIME;
          POPULATION:=0.0;
          DEVPATH:=NONE;
          LINK:=NIL {not a "ring"}
    END


END; {Restart}


PROCEDURE GOODBYE; {Closes output file(s), if opened}
BEGIN
  IF HARDCOPY
    THEN CLOSE(GRAPHILE);
  IF EGGOUT
    THEN CLOSE(GRAPHILE);
  WRITELN(CHR(29) );
  WRITELN(CHR(29) );
  WRITELN('End of Navel Orangeworm Simulation Program.' , CHR(29) );
  WRITE(CHR(29) );
  EXIT(PROGRAM)    {Program always ends here.}
END; {of Goodbye}


{Segment-able Procedures above; Non-segment-able Procedures below}


PROCEDURE HELP; {Explains Command Mode prompt-line}

  VAR INFILE:TEXT;
      LINEDATA:STRING;

  BEGIN
    IF TESTFILE('HELP.TEXT')
    THEN
      BEGIN
        RESET(INFILE,'HELP.TEXT');
        WRITE(CHR(11));
        WHILE NOT EOF(INFILE) DO
          BEGIN
            READLN(INFILE,LINEDATA);
```

```
          END;
        END
      ELSE WRITE(CHR(7),'Information file is not on disk !');
      CLOSE(INFILE);
    END; {of Help}


  OCEDURE UPDATE;
{Updates Pop[1 .. 6] to reflect inter-instar transitions for Init_pop &
  detailed output option}

VAR FUTUREDDS:REAL; {Deg-Days betw/ current-time and future LtoP event (E^) }
    E:EVENTPTR;
    NEWINSTAR:INTEGER;
    LASTIME:REAL; {time of previous Lartopup event found in loop below}

BEGIN { U p d a t e }
  IF (DDSBETWEEN(LASTUPDATE,CURRENTIME,{above} LARVATHR ) > 0.0)
   THEN {current pop is obsolete}
    BEGIN
      FOR NEWINSTAR:=1 TO 6 DO POP[NEWINSTAR]:=0.0;   {Leave Pop[-1] alone}

      IF ( COUNT[LARTOPUP] > 0 ) THEN {there are some larva around}
        BEGIN
          LASTIME:=CURRENTIME;
          FUTUREDDS:=0.0;

          E:=CURRENTEVENT;
          WHILE (E^.KIND <> FINAL) DO
            BEGIN
              IF (E^.KIND = LARTOPUP) THEN
                BEGIN
                  FUTUREDDS:=FUTUREDDS + DDSBETWEEN(LASTIME,E^.TIME,LARVATHR);
                  LASTIME:=E^.TIME;

                  {Find Newinstar}
                  NEWINSTAR:=0;
                  REPEAT
                    NEWINSTAR:=NEWINSTAR + 1
                  UNTIL ( FUTUREDDS >= MINREMIN[NEWINSTAR] );

                  POP[NEWINSTAR]:=POP[NEWINSTAR]  +  E^.POPULATION

                END; {If E^.kind = Lartopup}

              E:=E^.LINK
            END {While loop}
        END {If Count > 0}
    END;   {If "New"d-ds > 0.0}

  LASTUPDATE:=CURRENTIME

END; { U p d a t e }


PROCEDURE MOREHELP; {Explains Newdata prompt line}

  VAR   INFILE:TEXT;
        LINEDATA:STRING;

  BEGIN
    IF TESTFILE('MOREHELP.TEXT')
    THEN
      BEGIN
        RESET(INFILE,'MOREHELP.TEXT');
```

```
        WHILE NOT EOF(INFILE) DO
          BEGIN
            READLN(INFILE,LINEDATA);
            WRITELN(LINEDATA);
          END;
      END
    ELSE WRITE(CHR(7),'Information file is not on disk !');
    CLOSE(INFILE);
  END; {Morehelp in Newdata}



PROCEDURE EGGTRAP;

  VAR TIMES: ARRAY[0..15] OF REAL;
      SAMPLECOUNT,I:INTEGER;
      FIRSTDROP:REAL;
      HARDNAME:STRING;
      T:EVENTPTR;
      CH:CHAR;

  BEGIN
    EGGOUT:=TRUE;
    REPEAT
      WRITE(CHR(12),'How many samples taken ? ');
      READLN(SAMPLECOUNT)
    UNTIL SAMPLECOUNT > 0;
    REPEAT
      WRITE(CHR(13),'Time of first emplacement (Julian day) ? ');
      READLN(FIRSTDROP);
    UNTIL FIRSTDROP >= CURRENTIME ;
    WRITE(CHR(13),'Weekly sampling ? ');
    READ(CH);
    WRITELN(CHR(13));
    IF CH IN ['Y','y']
    THEN FOR I:= 0 TO SAMPLECOUNT DO
           BEGIN
             TIMES[I]:=FIRSTDROP;
             FIRSTDROP:=FIRSTDROP + 7.0;
           END
    ELSE  BEGIN
             TIMES[0]:=FIRSTDROP;
             FOR I:= 1 TO SAMPLECOUNT DO
               BEGIN
                 IF I < SAMPLECOUNT
                 THEN WRITE('Time of retrieval #',I,'  ,  emplacement #',I+1,' ?
                 ELSE WRITE('Time of last retrieval ? ');
                 READLN(TIMES[I]);
               END;
             END;
    FOR I:=0 TO SAMPLECOUNT DO
      BEGIN
        T:=NEWPTR;
        WITH T^ DO
          BEGIN
            KIND:=EGGSAMPLE;
            TIME:=TIMES[I];
            POPULATION:=MINPOP;
            DEVPATH:=NONE;
            LINK:=NIL;
          END;   { with }
        ADD(T);
        IF T=NIL
        THEN  BEGIN
```

```
  THEN
    BEGIN
      IF SYM[1]<>' '
      THEN
        BEGIN
          WRITE(CHR(13),'Change scale factor of eggs ?    ' );
          READ(CH);
          IF CH IN ['Y','y']
          THEN
            BEGIN
              WRITE(CHR(13),'Enter new scale factor ->    ');
              READLN(SFEGGS);
            END
          ELSE WRITELN;
        END;
      IF SYM[2]<>' '
      THEN
        BEGIN
          WRITE(CHR(13),'Change scale factor for larvae stage #1  ?    ');
          READ(CH);
          IF CH IN ['Y','y']
          THEN
            BEGIN
              WRITE( CHR(13),'Enter new scale factor ->    ');
              READLN(SFLARVA1);
            END
          ELSE WRITELN;
        END;
      IF SYM[3]<>' '
      THEN
        BEGIN
          WRITE(CHR(13),'Change scale factor for total larvae population ?    ');
          READ(CH);
          IF CH IN ['Y','y']
          THEN
            BEGIN
              WRITE( CHR(13),'Enter new scale factor ->    ');
              READLN(SFTOTLARVA);
            END
          ELSE WRITELN;
        END;
      IF SYM[4]<>' '
      THEN
        BEGIN
          WRITE(CHR(13),'Change scale factor for adults ? ');
          READ(CH);
          IF CH IN ['Y','y']
          THEN
            BEGIN
              WRITE( CHR(13),'Enter new scale factor');
              READLN(SFADULTS);
            END;
        END;
    END;
END;


PROCEDURE OUTPUTYPE;
VAR CH:CHAR;
    HARDNAME:STRING;
    T:EVENTPTR;

BEGIN              {of O u t p u t u p e}
```

```
                IF I=0 THEN WRITE('1st emplacement rejected')
                       ELSE WRITE('Emplacement #',I+1,' rejected');
              END;
      END;   { for }

    IF EGGOUT
    THEN  BEGIN
            WRITELN;
            REPEAT
              WRITE(CHR(13),'Printed to R(emote    P(rinter  ? ');
              READ(CH);
              WRITELN;
            UNTIL ( CH IN ['R','r','P','p'] );
            IF CH IN ['P','p'] THEN HARDNAME:='PRINTER:'
                               ELSE HARDNAME:='REMOTE:';
            {$I-}
            RESET(GRAPHILE,HARDNAME);
            IF ( IORESULT <> 0 )
            THEN BEGIN
                   EGGOUT:=FALSE;
                   WRITELN;
                   WRITELN(CHR(7),HARDNAME,' is not on line ! type <cr> to contir
                   READLN;
            {$I+}
                 END;
          END;
  END;


PROCEDURE WHATTOGRAPH;

 VAR CH:CHAR;
     I:INTEGER;

 BEGIN
   FOR I:=1 TO 4 DO SYM[I]:=' ';
   WRITE(CHR(12));
   WRITE(CHR(13),CHR(13),'Do you want egg population on graph ?  ');
   READ(CH);
   IF CH IN ['Y','y'] THEN SYM[1]:='e';
   WRITE(CHR(13),CHR(13),'Do you want larva stage #1 on graph ?  ');
   READ(CH);
   IF CH IN ['Y','y'] THEN SYM[2]:='1';
   WRITE(CHR(13),CHR(13),'Do you want the total larva population on the graph  ?
   READ(CH);
   IF CH IN ['Y','y'] THEN SYM[3]:='*';
   WRITE(CHR(13),CHR(13),'Do you want adult population on the graph ? ');
   READ(CH);
   IF CH IN ['Y','y'] THEN SYM[4]:='a';
 END;


PROCEDURE SCALEFACTORS;

 VAR CH:CHAR;

 BEGIN
   WRITE(CHR(12));
   WRITELN('The present scale factors are',CHR(13) );
   WRITELN('    ',SFEGGS:4:2,'  for eggs');
   WRITELN('    ',SFLARVA1:4:2,'  for larvae stage #1');
   WRITELN('    ',SFTOTLARVA:4:2,'  for total larvae population');
   WRITELN('    ',SFADULTS:4:2,'   for adult population');
   WRITE( CHR(13),'Change Scale factors (y/n) ?');
   READ(CH);
```

```
IF HARDCOPY THEN CLOSE(GRAPHILE);
EGGOUT:=FALSE;
HARDCOPY:=FALSE;
DUMP:=FALSE;
BASIC:=FALSE;
WRITE(CHR(12) );
WRITELN('Output:   ');
WRITELN;
WRITE('Singlestep (y/n) ? ');
READ(CH);
WRITELN;
IF ( CH  IN  ['Y','y'] ) THEN SINGLESTEP:=TRUE
                         ELSE SINGLESTEP:=FALSE;
WRITELN;
WRITE('Do you want basic output after each event? ');
READ(CH);
WRITELN;
IF ( CH  IN  ['Y','y'] ) THEN BASIC:=TRUE
                         ELSE BASIC:=FALSE;

WRITELN;
WRITE('Do you want a dump of the list after each event? ');
READ(CH);
WRITELN;
IF ( CH  IN  ['Y','y'] ) THEN DUMP:=TRUE
                         ELSE DUMP:=FALSE;

WRITE(CHR(13),'Do you want egg trapping output ? ');
READ(CH);
WRITELN;
IF CH IN ['Y','y'] THEN   EGGTRAP
                   ELSE   EGGOUT:=FALSE;

IF  NOT EGGOUT THEN BEGIN


  WRITELN;
  WRITE('Hardcopy? ');
  READ(CH);
  WRITELN;
  IF ( CH  IN  ['Y','y'] ) THEN HARDCOPY:=TRUE
                           ELSE HARDCOPY:=FALSE;

  IF HARDCOPY  THEN
      BEGIN

      WHATTOGRAPH;
      SCALEFACTORS;
      WRITE(CHR(12));

                PRINTSTART:=TRUNC(CURRENTIME/NEXTOUT)*NEXTOUT;
      IF ( PRINTSTART < CURRENTIME )
      THEN
              PRINTSTART:=PRINTSTART + NEXTOUT;

      T:=NEWPTR;
      WITH T^ DO BEGIN
                   KIND:=PAUSE;
                   TIME:=PRINTSTART;
                   POPULATION:=MINPOP;
                   DEVPATH:=NONE;
                   LINK:=NIL
                 END;
      ADD(T);
      IF ( T = NIL )
```

```
                    WRITELN('Initial Pause event rejected!')
                END
            ELSE BEGIN
                   WRITELN;
                   REPEAT
                     WRITE(CHR(13),'Printed To:  R(emote  or  P(rinter ? ');
                     READ(CH);
                     WRITELN
                   UNTIL ( CH  IN  ['P','p','R','r'] ) ;

                   IF ( CH  IN  ['P','p'] ) THEN HARDNAME:='PRINTER:'
                                            ELSE HARDNAME:='REMOTE:' ;
                   {$I-}
                   RESET(GRAPHILE,HARDNAME);
                   IF ( IORESULT <> 0 )
                     THEN BEGIN HARDCOPY:=FALSE;
                                WRITELN;
                                WRITELN(CHR(7), HARDNAME,' is not on line!');
                                READLN;
                          END
                   {$I+}
                 END
        END;
       END;
  WRITE( CHR(12) );
END; {of O u t p u t y p e }


PROCEDURE PURGER; {Clears aborted output-files}
BEGIN
  WRITELN(CHR(12),CHR(13),CHR(13),'PURGER.');

  IF EGGOUT
    THEN CLOSE(GRAPHILE);
   IF HARDCOPY
    THEN CLOSE(GRAPHILE)

END; {of Purger}


{$I#5:ONE} {Procedure ONEVENT is an include file called "ONE.TEXT".
          It is the "central" simulation procedure. }

{Definitions for non-segment-able Forward references have been made into
 an include file called "FWD.TEXT" as of 9-10-79  : }


{$I#5:FWD}


PROCEDURE DEBUG;
VAR I,J:INTEGER;
    K:EVENTKIND;
    E:EVENTPTR;
    R:REAL;

BEGIN
  IF SINGLESTEP THEN  UNITCLEAR(2);

  IF BASIC THEN
    BEGIN
      WRITELN;
      WRITELN;
      WRITELN('Basic output at ',CURRENTIME:7:2);
      WRITELN;
```

```
      WRITELN('Total degree days above the larval threshold ');
      WRITELN('accumulated from ',STARTIME:5:2,' is ',TOTALDDS:6:2);
      WRITELN;

      WRITELN('Pop:     Eggs     Larva      Pupa    Adults');
      WRITELN('     ',POP[0]:8:1,' ',POP[-1]:8:1,' ' , POP[7]:8:1 ,
                                                ' ' , POP[8]:8:1   );

      WRITELN;
      WRITE('Cohorts:  H    L    P    L    D    S    K',CHR(13),'         ');
      FOR K:=HATCH TO KNOCK DO WRITE(COUNT[K]:4);
      WRITELN;

      WRITELN;
      IF ( CURRENTEVENT <> NIL )
        THEN WRITELN('Ord of kind of next event is ' ,
                                                ORD(CURRENTEVENT^.KIND) )
        ELSE WRITELN('Last event. ');

      WRITELN
    END;

END;


BEGIN {D r i v e r}
  GOTOXY(0,0);
  WRITELN('Navel Orangeworm Simulation. ');

  INIT;

  REPEAT {Reset until Quit}

    RESTART;

    WRITE(CHR(12) );
    GOTOXY(0,2);
    WRITE(
    'If you need help, type "H" when the Command Mode prompt-line appears. ');


    REPEAT {Command Mode}
      GOTOXY(0,0);
     WRITE('Command Mode:  H(elp  N(ewdata  O(utput  S(imulate  R(eset  Q(uit');
      WRITELN(CHR(29), CHR(13), CHR(29),  CHR(7) );
      GOTOXY(67,0);
      NEEDHELP:=TRUE;
      READ(KEYBOARD,COMMAND);
      CASE COMMAND OF
        'N','n':BEGIN
                  NEEDHELP:=FALSE;
                  NEWDATA
                END;


        'S','s':BEGIN {Set up "asynchronous" simulation loop}

                  {CLOSE(OUTPUT);
                   REWRITE(OUTPUT,'PRINTER:');}
                  NEEDHELP:=FALSE;
                  HASBEGUN:=TRUE;
                  WRITE(CHR(12) );
                  WRITELN('Simulation in progress...  Type <sp> to interrupt. ');
                  WRITELN(CHR(29) );
```

```
                    UNITREADV(2,BREAK,1,1,1)  {asynchronous interrupt}

                    WHILE (UNITBUSY(2) AND CONT) DO {Asynchronous interrupt loop}
                      BEGIN

                        ONEVENT;

                        COUNT[CURRENTEVENT^.KIND]:=COUNT[CURRENTEVENT^.KIND] - 1;

                        TEMPFREE:=FREE;
                        FREE:=CURRENTEVENT;
                        CURRENTEVENT:=CURRENTEVENT^.LINK;
                        FREE^.LINK:=TEMPFREE;

                        IF ( CURRENTIME > OLDTIME )
                        THEN                {update the degree day count}
                           BEGIN
                        TOTALDDS:=TOTALDDS + DDSBETWEEN(OLDTIME,CURRENTIME,LARVATHF
                        OLDTIME:=CURRENTIME
                           END;

                        DEBUG

                      END; {of Asynchronous interrupt loop}

                    WRITELN(CHR(29) ,CHR(7) );
                    IF CONT
                      THEN
                        BEGIN
                          WRITELN('Simulation interrupted at ',
                                  CURRENTIME:7:2 ,
                                  ', model-time.' , CHR(29) );

                          WRITELN(CHR(29) )
                        END
                      ELSE
                        BEGIN
                          WRITELN('Simulation stopped at ',
                                  CURRENTIME:7:2 ,
                                  ' model time.' , CHR(29) );

                          (*GOODBYE {& exit program} *)

                          COMMAND:='R'

                        END
                  {CLOSE(OUTPUT);
                   REWRITE(OUTPUT,'CONSOLE:');}
                  END; {case of Ch="s" => Simulate}


        'O','o':BEGIN
                  NEEDHELP:=FALSE;
                  OUTPUTYPE
                END;


'?','R','q','r':NEEDHELP:=FALSE;

        'H','h':NEEDHELP:=TRUE

      END; {Case Command of}


      IF (NEEDHELP) THEN HELP
```

```
    UNTIL (COMMAND IN ['Q','q','R','r'] );
    IF (COMMAND IN ['R','r'] ) THEN PURGER
  UNTIL (COMMAND IN ['Q','q'] );
  GOODBYE
END. {D r i v e r}
```

```
{This is MAIN.TEXT, an include file for NOW.TEXT as of Sep.17-80 }

CONST DEFAULT='TEMPS' ;

      DEFSTART=0.0;
      MAXTIME=729.5;       { 2 years, ending at noon of the last day }
      DEFHARVEST=227.0;    {Default harvest time for cash crop  }
      DEFPHARVEST=248.0;   {Default harvest time for pollinators}
      TEMPPTS=730;         {Single year of min-max temperatures will
                               be repeated for the 2nd year    }

      DELTA=0.0833;   {Lay events within 2 hrs. of eachother are merged}

      {Insecticide spray constants: }

      IEFFLONL=0.90; {Initial EFFectiveness of Larvacidal ON Larva}
      IEFFAONA=0.90; {Initial EFFectiveness of Adulticidal ON Adults}
      DURLARSP=21.0; {DURation of LARvacidal SPray}
      DURADLSP=21.0; {DURation of ADuLticidal SPray}


      {Tree constants: }

           {Area of plot = 1.0 acre}

           {Cash Crop: }
           CSHTREES=70.0;     {Number of cash crop trees in model plot}
           BEGCRACK=190.0;  {BEGin-hull-CRACK @ Jul-9 = 2% }
           ENDCRACK=213.0;  {END-hull-CRACK @ Aug-1 = 99%  }
           IMUMMYPT=20.0;   {Initial no. of MUMMY's Per Tree (on Jan. 1 ) }
           NUTSPT=2000.0;   {Number of NUTS Per Tree}

           {Pollinators: }
      {?}  POLTREES=20.0;    { (VAR) CSHOVERP = CSHTREES / POLTREES }
      {?}  PBEGCRACK=170.0;
      {?}  PENDCRACK=193.0;
      {?}  PMUMMYPT=20.0;
      {?}  PNUTSPT=2000.0;


      {Insect development constants: }

           {Development times: }

           DDINEGG=108.0 ; {"DD" = Degree-Days (metabolic time) }
           DDIN1=165.0 ; {First of 6 larval instars}
           DDIN2=165.0 ;
           DDIN3=165.0 ;
           DDIN4=165.0 ;
           DDIN5=165.0 ;
           DDIN6=165.0 ; {Sum of 1 thru 6 is (VAR)  DDINLARVA}
```

```
          DDINADLT=216.0 ;

          {Development path selection preferences: }

               {Larval infestation: }
        {?}    PRFMTBTW=0.75; {PReFer MeaT to BeTWeen-hull-&-meat }

          {Survival rates for different branches of the development path: }

             {Eggs: }
             HLFRHTCH=0.85 ; {for HulL, FRaction that HaTCH}
             TMFRHTCH=0.85 ; {for Tree Mummy, FRaction that HaTCH}
           (* "GMFRHTCH=0.00"; {for Ground Mummy, FRaction that HaTCH}   *)

             {Larva: }
        {?}  MTFRPUPT=0.95 ; {for MeaT, FRaction that PUPaTe}
        {?}  BTFRPUPT=0.92 ; {for BeTw, FRaction that PUPaTe}
        {?}  TMFRPUPT=0.90 ; {for Tree Mummy, FRaction that PUPaTe}
             GMFRPUPT=0.20 ; {for Ground Mummy, FRaction that PUPaTe}

             {Pupa: }
             MTFRRMRG=0.98 ; {for MeaT, FRaction that Re-eMeRGe}
             BTFRRMRG=0.98 ; {for BeTw, FRaction that Re-eMeRGe}
             TMFRRMRG=0.98 ; {for Tree Mummy, FRaction that Re-eMeRGe}
             GMFRRMRG=0.10 ; {for Ground Mummy, FRaction that Re-eMeRGe}

             {Adults: }
             ADLTLIVE=0.75;

          {Egg laying constants: }

          EGGSPERFEM=85.0; {EGGS PER FEMale}
          LAYTIME=0.9375 ; {time of day when eggs are laid=22:30 }
    {?}   PEAKDDS=63.0 ;   {D-D's accumulated up to PEAK in oviposition rate}


TYPE EVENTKIND=(HATCH, LARTOPUP, PUPTOADLT, LAY, DEATH,
                SPRAY, KNOCK, SWEEP, PAUSE, EGGSAMPLE, FINAL ); {4 bits}

     PATH=(HULL,   MEAT,   BETWEEN,   TREEMUMMY,   GROUNDMUMMY,   {Cash crop}
           PHULL, PMEAT, PBETWEEN, PTREEMUMMY, PGROUNDMUMMY, {Pollinators}
           LARVACIDE, ADULTICIDE,   NONE ); {4 bits}

     EVENTPTR=^EVENT;

     EVENT = RECORD
                KIND : EVENTKIND;
                TIME : REAL;
          POPULATION : REAL;
             DEVPATH : PATH;
                LINK : EVENTPTR
           END;


VAR COMMAND, BREAK: CHAR;
    {DAYEGGS, }                {Used only for DAILY egg output graphing in LAYER}
    CONT,
    NEEDHELP,
    HASBEGUN,
    SINGLESTEP,
    BASIC,
    DUMP,
    HARDCOPY,
    FIRSTRAP,
```

```
    TEMPS:FILE OF REAL;
    GRAPHILE:INTERACTIVE;
    TEMPNAME:STRING;
    NODEFAULT:BOOLEAN;

    TEMP:ARRAY[0..1459] OF REAL;  {Temporary fix only}
    SYM :ARRAY[1..4] OF CHAR;

    SFEGGS,
    SFLARVA1,
    SFTOTLARVA,
    SFADULTS:REAL;

    MINPOP,           {   Population STRICTLY below this are ignored  }
    EGGTHR,
    LARVATHR,
    PUPATHR,
    ADULTTHR,
    CALLTHR:REAL;     {    Mating and oviposition threshold  }

    HEAP:^INTEGER;
    CURRENTEVENT,FREE,TEMPFREE:EVENTPTR;

    STARTIME,
    STOPTIME,
    HARVESTIME,  {Also used as turnover point for (cash crop) mummy drop}
    PHARVESTIME, {Also used as turnover point for (pollinator) mummy drop}
    CURRENTIME : REAL;

    NEXTOUT:REAL; {Spacing between output events (in days) }
    PRINTSTART:REAL;  {Start time for PAUSE events i.e. graphing}

    COUNT:ARRAY[HATCH .. FINAL] OF INTEGER; {Count of each event-type in list }
    POP:ARRAY[-1 .. 8] OF REAL; {Detailed population count made current
                                          by the procedure, Update}

    LASTUPDATE:REAL; {time of last update of Pop[1..6]
                      to reflect inter-instar transitions}



    MINREMIN:ARRAY[0 .. 6] OF REAL; {Update uses 1 .. 6;
                                     Schedule (in Init_pop) uses 0 .. 5}

    DDINLARVA:REAL; {Sum of six larval instar development times}

    CSHOVERP:REAL;   { = CaSHTREES / POLlinator_TREES }

    OLDTIME:REAL;    {Last update of degree day count-see DRIVER in 'S'  }
    TOTALDDS:REAL;   {Degree days (above LARVATHR) from STARTIME-see DRIVER}

    EGGTRAPCOUNT:REAL;

    LASTPRINT:INTEGER;


{Dirty Tricks:  Chr(11) = erase between cursor and end of screen
                Chr(12) = form feed (clearscreen)
                Chr(13) = carriage return
                Chr(29) = erase between cursor and end of line

                Only the first 8 characters of the long names are significant.

                In Laywhile and Ddsbetween, some of the value parameters are
                  used as local variables. }
```

```
{Non-segment-able forward references: }

{The definitions corr. to these are in the Include file--"FWD.TEXT" }
FUNCTION SENSEFILE(NAME:STRING) : BOOLEAN; FORWARD;
PROCEDURE READFILE; FORWARD;

PROCEDURE ADDLAY(VAR P:EVENTPTR); FORWARD;
PROCEDURE ADD(VAR P:EVENTPTR); FORWARD;
FUNCTION NEWPTR : EVENTPTR; FORWARD;

FUNCTION AREA(A,B:REAL):REAL; FORWARD;
FUNCTION DDSBETWEEN(TO,T1, {above} THR : REAL) : REAL; FORWARD;
FUNCTION WHEN(DELDEGDAY, {above} THR : REAL ) : REAL; FORWARD;

FUNCTION HULLCRACK(T:REAL) : REAL; FORWARD;
FUNCTION MUMMYDROP(T:REAL) : REAL; FORWARD;
FUNCTION PMUMMYDROP(T:REAL) : REAL; FORWARD;

FUNCTION FEC(DDS:REAL) : REAL; FORWARD; {cumulative FECundity}
PROCEDURE LAYWHILE(FEMALES,CT,DEATHTIME,CUMDDS:REAL); FORWARD;
FUNCTION TESTFILE(NAME:STRING):BOOLEAN;FORWARD;

{The following are in NOW but called first in the include file NEWDATA}
PROCEDURE UPDATE; FORWARD;
PROCEDURE MOREHELP; FORWARD;
```

```
{This is FWD.TEXT, an include file for "NOW.TEXT" }

{Definitions for non-segment-able Forward references: }
FUNCTION TESTFILE  { :Boolean };

  VAR INFILE:TEXT;

  BEGIN
    {$I-}
    RESET(INFILE,NAME);
    IF ( IORESULT <> 0 ) THEN TESTFILE:=FALSE
                         ELSE TESTFILE:=TRUE;
    {$I+}
    CLOSE(INFILE);
  END;



FUNCTION SENSEFILE  { :Boolean }  ;      { (Forward referenced) }
VAR TESTFILE:FILE OF REAL;

BEGIN
  {$I-}
  RESET(TESTFILE,CONCAT(NAME,'.TEMP') );
  SENSEFILE:=( IORESULT = 0 );
  CLOSE(TESTFILE)
  {$I+}
END; {of Function Sensefile (Forward Referenced) }

{This procedure has only been patched up temporarily}

PROCEDURE READFILE;  { (Forward referenced) }
VAR I:INTEGER;

BEGIN
  WRITELN('Reading daily temperature data...');

  I:=-1;
  WHILE ( NOT(EOF(TEMPS) ) AND (I < TEMPPTS-1) ) DO
    BEGIN
      I:=I+1;
      TEMP[I]:=TEMPS^;
      TEMP[I+730]:=TEMPS^;
      GET(TEMPS);
    END;

  IF ( I < TEMPPTS-1 )
    THEN BEGIN
           WRITELN('WARNING!  There were only ',I DIV 2,' days of temp. data!');
           WRITELN('Type <ret> to continue.' , CHR(7) );
           READLN;
           WRITELN
         END
```

```
PROCEDURE ADDLAY  {  (Var P:Eventptr)  }  ;  { (Forward referenced) }
VAR SEARCH,
    EXACTIME,                {Currently (11-25) Addlay is ONLY called by Add() }
    NEIGHBOR:EVENTPTR;
    T:REAL;

BEGIN
  IF ( CURRENTEVENT^.KIND = FINAL )
    THEN BEGIN {Insert P^ at front}
           P^.LINK:=CURRENTEVENT;
           CURRENTEVENT:=P
         END
    ELSE BEGIN
           T:=P^.TIME;
           SEARCH:=CURRENTEVENT;
           EXACTIME:=NIL;
           NEIGHBOR:=NIL;

           WHILE ( (SEARCH^.TIME <= (T + DELTA))
                              AND
                   (SEARCH^.KIND <> FINAL)        ) DO
             BEGIN
               IF (EXACTIME = NIL)
                 THEN BEGIN
                        IF (SEARCH^.LINK^.TIME > T) {Search^ is LAST of <= time}
                          THEN EXACTIME:=SEARCH
                      END;
               IF (  ABS(SEARCH^.TIME - T)  <= DELTA )
                 THEN BEGIN
                        IF (       (SEARCH^.KIND = LAY)
                                          AND
                           (SEARCH^.DEVPATH = P^.DEVPATH) ) {i.e.  compatible}
                          THEN NEIGHBOR:=SEARCH
                      END;

               SEARCH:=SEARCH^.LINK

             END; {While}

           IF ( NEIGHBOR <> NIL )
             THEN BEGIN {merge P^ into Neighbor^  }
                    NEIGHBOR^.POPULATION:=NEIGHBOR^.POPULATION + P^.POPULATION;

                    P^.LINK:=FREE;
                    FREE:=P
                  END
             ELSE BEGIN
                    IF ( EXACTIME <> NIL )
                      THEN BEGIN {Insert P^ after Exactime^  }
                             P^.LINK:=EXACTIME^.LINK;
                             EXACTIME^.LINK:=P;
                             COUNT[P^.KIND]:=COUNT[P^.KIND] + 1
                           END
                      ELSE {i.e.  the while loop was not executed be-
                                    cause Currentevent^.Time > (T+d)  }
                           BEGIN {insert @ beginning}
                             P^.LINK:=CURRENTEVENT;
                             CURRENTEVENT:=P;
                             COUNT[P^.KIND]:=COUNT[P^.KIND] + 1
                           END
                  END

         END
```

```
PROCEDURE ADD  { (Var P:Eventptr) }  ;  { (Forward referenced) }
{Inserts events pointed to by P into event list in order of Time and
 updates the global Count array.  (P comes back nil means event not added.}

VAR T,S:EVENTPTR;

.GIN

  (*WRITELN;
  WRITELN({'C^.Time = ',C^.TIME:7:2,}'Currentime = ',CURRENTIME:7:2);
  WRITELN('Time of event to be added is ',P^.TIME:7:2);
  WRITELN('Population of cohort is ',P^.POPULATION:8:1);
  WRITELN('Ord(Eventkind) = ',ORD(P^.KIND) );
  WRITE('Type <cr> to continue ');
  READLN;
  WRITELN; *)

  IF ( (P^.TIME >= MAXTIME)      {NOTE:  NO event can occur AT Maxtime or later}
             OR
       (P^.TIME < CURRENTIME)
             OR
       (P^.POPULATION < MINPOP) )

    THEN                        {D i s c a r d  P^}
         BEGIN
           { WRITELN( 'Event of kind ',ORD(P^.KIND),' discarded.'); }
           TEMPFREE:=FREE;
           FREE:=P;
           FREE^.LINK:=TEMPFREE;
           P:=NIL
         END

    ELSE                        {I n s e r t  P^  (somewhere) }
       BEGIN
         IF ( P^.KIND = LAY ) THEN ADDLAY(P)
         ELSE BEGIN
           IF ( P^.TIME < CURRENTEVENT^.TIME )
             THEN BEGIN {Insert P^ at beginning of list}
                    T:=CURRENTEVENT;
                    CURRENTEVENT:=P;
                    CURRENTEVENT^.LINK:=T;
                    COUNT[P^.KIND]:=COUNT[P^.KIND] + 1
                  END
             ELSE BEGIN
                    IF ( P^.TIME = CURRENTEVENT^.TIME )
                      THEN
                        BEGIN {Insert P^ immediately after Currentevent}
                              { (since cause usually precedes effect)  }
                          S:=CURRENTEVENT^.LINK;
                          CURRENTEVENT^.LINK:=P;
                          P^.LINK:=S;
                          COUNT[P^.KIND]:=COUNT[P^.KIND] + 1
                        END
                      ELSE
                        BEGIN {Insert P^ strictly between first and last events}
                           {Find LAST correct spot: }
                           T:=CURRENTEVENT;
                           WHILE ( P^.TIME  >=  T^.LINK^.TIME ) DO T:=T^.LINK;

                           {Now insert P^ immediately after T^}
                           S:=T^.LINK;
                           T^.LINK:=P;
                           P^.LINK:=S;
                           COUNT[P^.KIND]:=COUNT[P^.KIND] + 1
```

```
                    END
          END {else not a lay event}
        END
END; {Add}


FUNCTION NEWPTR  { :Eventptr }  ;      { (Forward referenced) }
R FRIG:EVENTPTR;
BEGIN
  IF ( FREE = NIL )
    THEN BEGIN NEW(FRIG);
               NEWPTR:=FRIG
         END
    ELSE BEGIN NEWPTR:=FREE;
               FREE:=FREE^.LINK
         END
END; {of Function Newptr  (Forward referenced) }

FUNCTION AREA  { (A,B:REAL):REAL }  ;  { (Forward referenced) }
{Computes area above an interval of length 0.5 on the x-axis and below
 the straight line joining points at heights A and B (which may be positive,
 negative, or zero). Note that if the interval length is changed to b units
 then the corresponding area becomes 2b*AREA(A,B).
 This function is used in the calculation of degree day accumulations above
 a threshold, using linear temperature interpolation between daily min-max
 values.}

BEGIN
  IF ( A > 0.0 )
    THEN BEGIN
         IF (B > 0.0 )
           THEN AREA:=(A+B) / 4.0      {trapezoid}
           ELSE AREA:=A*A/(4.0*(A-B)) {triangle}
         END
    ELSE BEGIN
           IF (B > 0.0)
              THEN AREA:=B*B/(4.0*(B-A))    {triangle}
              ELSE AREA:=0.0       {below x-axis}
          END;
END; {Area}


FUNCTION DDSBETWEEN (*(TO,T1, {above} THR:Real) : Real*) ;  { (Forward ref.'d) }
{Computes the Deg-Day'S BETWEEN TO and T1 above Thr using Area }

VAR A,B,AREAS:REAL;
    KO,K1:INTEGER;

BEGIN
   KO:=TRUNC( 2.0 * TO );
   K1:=TRUNC( 2.0 * T1 );

   A:= TEMP[KO] + ( 2.0 * TO - KO ) * ( TEMP[KO+1] - TEMP[KO]) - THR;

   IF( T1 <= (( KO+1.0 ) * 0.5 ))   {TO, T1 are in the same half day}
       THEN BEGIN

         B:=TEMP[KO] + ( 2.0 * T1 - KO ) * (TEMP[KO+1]-TEMP[KO]) - THR;
         DDSBETWEEN:=2.0 * ( T1 - TO ) * AREA( A,B )

       END
       ELSE BEGIN

         B:=TEMP[KO+1] - THR;
         AREAS:=( KO + 1.0 - 2.0 * TO ) * AREA( A,B );
```

```
          WHILE ( KO+1 < K1 ) DO
            BEGIN

              KO:=KO + 1;
              A:=B;
              B:=TEMP[KO+1] - THR;
              AREAS:=AREAS + AREA( A,B )

            END;
          IF ( T1 <= (K1* 0.5) ) THEN  DDSBETWEEN:=AREAS
           ELSE BEGIN

             A:=B;
             B:=TEMP[K1] + ( 2.0 * T1 - K1 ) * ( TEMP[K1+1] - TEMP[K1] ) - THR;

             DDSBETWEEN:=AREAS + ( 2.0 * T1 - K1 ) * AREA( A,B )

           END
      END;

END; {of Function Ddsbetween  (Forward Referenced) }


FUNCTION WHEN  (*(DELDEGDAY, {above} THR:Real) : Real*) ;  { (Forward ref.'d) }
{Computes the (calendar) time after T=Currentime when Deldegday
 d-d's have been accumulated above the threshold temp., THR, assuming
 that the temperature is linear between daily max. and min. values.    }

VAR K:INTEGER;
    A,B,           {Trapezoid heights}
    L,             {Trapezoid base length}
    AREAS,         {Cumulative trapezoid areas}
    LFTOVER,       {Degree days still to be accumulated, i.e. LeFTOVER}
    BST,           {BaSeTime, where interpolation begins}
    DELT:REAL;     {Incremental time to be added to BaSeTime}

BEGIN

LFTOVER:=DELDEGDAY;

BST:=CURRENTIME;
K:=TRUNC(2.0*CURRENTIME);

A:=TEMP[K] + (2.0*BST-K)*(TEMP[K+1] - TEMP[K])-THR;
B:=TEMP[K+1] - THR;
AREAS:=(K+1- 2.0*BST)*AREA(A,B);

WHILE ( (AREAS < DELDEGDAY) AND (K<2*(TEMPPTS-1)) ) DO
   BEGIN
     K:=K+1;
     LFTOVER:=DELDEGDAY - AREAS;
     BST:=0.5*K;

     A:=B;
     B:=TEMP[K+1]-THR;
     AREAS:=AREAS + AREA(A,B)
   END;

   ( (AREAS < DELDEGDAY) AND (K >= 2*(TEMPPTS-1)))

    THEN
         WHEN:=MAXTIME

    ELSE BEGIN
          L:=(K+1-2.0*BST)*0.5;
```

```
            THEN

               DELT:=(SQRT(LFTOVER*(B-A)*2.0*L)-A*L)/(B-A)

            ELSE

               DELT:=2.0*LFTOVER/(A+SQRT(A*A+2.0*LFTOVER*(B-A)/L));

            WHEN:=BST + DELT
   END
END; {of Function When  (Forward Referenced) }


FUNCTION HULLCRACK  { (T:Real) : Real  } ;  { (Forward Referenced) }
BEGIN
  IF (T < BEGCRACK)
    THEN HULLCRACK:=0.0
    ELSE BEGIN
           IF (T >= ENDCRACK)
             THEN HULLCRACK:=0.99
             ELSE HULLCRACK:=0.02 + (((T-BEGCRACK) / (ENDCRACK-BEGCRACK))* 0.97)
         END
END; { Hullcrack }


FUNCTION MUMMYDROP  { (T:Real) : Real  } ;  { (Forward Referenced) }
{Mummydrop(t) = FRACTION of tree-mummies that HAVE dropped
                            off cash crop trees since last cash harvest }
BEGIN
  IF ( T < HARVESTIME )
    THEN MUMMYDROP:=(1.0 / 365.0) * (365.0 - HARVESTIME + T)
    ELSE MUMMYDROP:=(1.0 / 365.0) * (T - HARVESTIME)
END; { Mummydrop }

FUNCTION PMUMMYDROP  { (T:Real) : Real  } ;  { (Forward Referenced) }
{Pmummydrop(t) = FRACTION of tree-mummies that HAVE dropped
                            off pollinator trees since last pollinator harvest }
BEGIN
  IF ( T < PHARVESTIME )
    THEN PMUMMYDROP:=(1.0 / 365.0) * (365.0 - PHARVESTIME + T)
    ELSE PMUMMYDROP:=(1.0 / 365.0) * (T - PHARVESTIME)
END; { Pmummydrop }


FUNCTION FEC { (DDS:REAL) : REAL } ;  { (Forward Referenced) }

{Cumulative eggs laid per unit female  (Current version ignores Devpath) }

BEGIN
  FEC:=EGGSPERFEM * ( 1.0 - EXP(-0.5 * ( (DDS/PEAKDDS)*(DDS/PEAKDDS) )) )
END;


PROCEDURE LAYWHILE  { (FEMALES,CT,DEATHTIME,CUMDDS:REAL) }  ;
VAR NEWDDS,
    EGGSLAID,NEWEGGS,                                  { ^  (Forward Referenced) }
    TMPOP,HLPOP:REAL;

    LAYDAY,DAY,LOWTIME:INTEGER;

    {F:INTERACTIVE;}      {Only used to check lowtimes assignments}

    T:EVENTPTR;
```

```
    DAY:=TRUNC(CT) + 1;
    IF (CT+1-DAY) < LAYTIME THEN LAYDAY:=DAY  ELSE LAYDAY:=DAY+1;
    NEWDDS := DDSBETWEEN(CT,(LAYDAY+LAYTIME-1), {above} ADULTTHR );
    EGGSLAID:= FEC(CUMDDS) ;

    WHILE ( ((LAYDAY + LAYTIME-1) < DEATHTIME)
                          AND
            ((LAYDAY + LAYTIME) < MAXTIME)    ) DO
      BEGIN
        NEWEGGS:=FEMALES * FEC(CUMDDS + NEWDDS);
        IF  LAYTIME > 0.5    THEN LOWTIME:=2*LAYDAY
                        ELSE LOWTIME:=2*LAYDAY-2;

          {REWRITE(F,'PRINTER:');
          WRITELN(F,'Currentime is ',CT:7:2,' Layday is ',LAYDAY:7);
          WRITELN(F,' Lowtime is ',LOWTIME:7);
          WRITELN(F,'Temperature at Lowtime is ',TEMP[LOWTIME]:7:2);
          WRITELN(F);
          WRITELN(F);
          CLOSE(F);}

          IF (TEMP[LOWTIME] >= CALLTHR) THEN
          BEGIN
            TMPOP:=( ((1.0-MUMMYDROP(LAYDAY))*IMUMMYPT) /
             (HULLCRACK(LAYDAY)*NUTSPT + ((1.0-MUMMYDROP(LAYDAY))*IMUMMYPT)) ) *
                                          ( NEWEGGS - EGGSLAID ) ;
            HLPOP:=(NEWEGGS - EGGSLAID) - TMPOP ;

            T:=NEWPTR;
            WITH T^ DO BEGIN KIND:=LAY;
                             TIME:=LAYDAY + LAYTIME-1;
                             POPULATION:=HLPOP;
                             DEVPATH:=HULL;
                             LINK:=NIL
                       END;
            ADD(T);

            T:=NEWPTR;
            WITH T^ DO BEGIN KIND:=LAY;
                             TIME:=LAYDAY + LAYTIME-1;
                             POPULATION:=TMPOP;
                             DEVPATH:=TREEMUMMY;
                             LINK:=NIL
                       END;
            ADD(T);

            {Leave Pop[] alone}

            EGGSLAID:=NEWEGGS
          END;
        CUMDDS:=CUMDDS + NEWDDS;
        NEWDDS:=DDSBETWEEN( (LAYDAY+LAYTIME-1), (LAYDAY+LAYTIME), ADULTTHR);
        LAYDAY:=LAYDAY + 1
      END {While loop}

END; {Laywhile}
```

```
                    WRITE(P^.TIME:7:2);   ');
                    IF ( P^.DEVPATH = LARVACIDAL)
                      THEN WRITELN('Larvacidal')
                      ELSE WRITELN('Adult-effective')
                  END;
              P:=P^.LINK
            END; {While P thru list}
      END {Then Count>0}
    ELSE {There are no sprays scheduled yet.}
      WRITELN('None');

  REPEAT
    WRITE(CHR(13), 'Add a spray event? ');
    REPEAT
      READ(KEYBOARD,CH)
    UNTIL ( CH IN ['Y','y','N','n'] );
    IF ( CH IN ['Y','y'] )
      THEN BEGIN
             WRITE(CHR(13),'When? ');
             READLN(SPRAYTIME);
             IF (SPRAYTIME >= CURRENTIME)
               THEN
                 BEGIN
                   WRITE('L(arvacidal or A(dult-effective? ');
                   REPEAT
                     READ(CH)
                   UNTIL (CH IN ['L','l','A','a'] );

                   P:=NEWPTR;
                   P^.KIND:=SPRAY;
                   P^.TIME:=SPRAYTIME;
                   P^.POPULATION:=MINPOP;
                   IF (CH IN ['L','l'] )
                     THEN P^.DEVPATH:=LARVACIDAL
                     ELSE P^.DEVPATH:=ADULTICIDAL;

                   ADD(P);

                   IF ( P <> NIL )
                     THEN WRITELN(CHR(13),'Spray event added.')
                     ELSE WRITELN(CHR(13),'Spray event too late to be added!')
                 END
               ELSE
                 BEGIN
                   WRITELN('You cannot schedule a spray for "the past".');
                   WRITELN('Current model time is ',CURRENTIME:7:2)
                 END
           END {Ch="y" => add}
  UNTIL ( CH IN ['N','n'] )   {i.e. no more additional sprays}

END; {Setspray}

PROCEDURE SETKNOCK;
BEGIN
  WRITE(CHR(12) );
  WRITE('Knock is not yet implemented.  Type <cr> to continue.');
  READLN;
  WRITELN;
  {Setknock is very much like setspray }
END; {Setknock}


BEGIN                {of N e w d a t a }
  WRITE(CHR(12) );
  REPEAT
```

```
    WRITELN(
'Newdata:  M(ore_help  P(arameters  W(eather  I(nit_pop  S(pray  K(nock  Q(uit'
                                                                  ,CHR(7));
    {Eventually add:  "Remove mummies" (and perhaps "Harvest") }
    WRITE(CHR(29) );
    GOTOXY(79,0);

    NEEDMORE:=TRUE;
    READ(KEYBOARD,NEWCOMMAND);

    CASE NEWCOMMAND OF
      'P','p':BEGIN NEEDMORE:=FALSE;
                    NEWTIMES
              END;

      'W','w':BEGIN IF NOT ( HASBEGUN )
                      THEN BEGIN
                             NEEDMORE:=FALSE;
                             WEATHER
                           END
                      ELSE BEGIN
                             NEEDMORE:=FALSE;
                             WRITELN(CHR(12),CHR(13),CHR(13));
                             WRITELN('You cannot change the weather file ',
                                     'after starting the model run');
                             WRITELN('  nor after entering a new initial ',
                                     'population.' , CHR(7) )
                           END
              END;

      'I','i':BEGIN NEEDMORE:=FALSE;
                    INITPOP
              END;

      'S','s':BEGIN NEEDMORE:=FALSE;
                    SETSPRAY
              END;

      'K','k':BEGIN NEEDMORE:=FALSE;
                    SETKNOCK
              END;

      'M','m':NEEDMORE:=TRUE;

      'Q','q':NEEDMORE:=FALSE
    END; {Case Newcommand of}

    IF NEEDMORE THEN MOREHELP

  UNTIL (NEWCOMMAND IN ['Q','q'] )

END; {of N e w d a t a}
```

```
PROCEDURE ONEVENT; {Calls for one event in ring to be processed,
                       but does NOT increment ^Currentevent          }


{This procedure and all of its (internal) sub-procedures are
 an include file for "NOW.TEXT" (as of 9-11-79)  }


VAR C:EVENTPTR; {Local copy of Currentevent}


{Below are all of the actual simulation (i.e. event processing) routines: }

PROCEDURE HATCHER; {Hatch event generates Lartopup event}
VAR MEATPOP,
    BETWPOP,
    TREEPOP,
    GROUNDPOP,
    DEVTIME   : REAL; {Devtime is calendar time of larva to pupa event.}

    MDDT:REAL; {MummyDrop @ Dev.Time allowing for "hysteresis"}

    T : EVENTPTR;

BEGIN
  IF ( C^.DEVPATH = HULL )
    THEN
      BEGIN
        POP[0]:=POP[0] - C^.POPULATION; {decrement egg pop. }

        MEATPOP:=C^.POPULATION * PRFMTBTW;
        BETWPOP:=C^.POPULATION - MEATPOP;
        MEATPOP:=MEATPOP * MTFRPUPT;              {larval mortality}
        BETWPOP:=BETWPOP * BTFRPUPT;

        DEVTIME:=WHEN(DDINLARVA, {above} LARVATHR);
        T:=NEWPTR;
        WITH T^ DO BEGIN KIND:=LARTOPUP;
                         TIME:=DEVTIME;
                         POPULATION:=MEATPOP;
                         DEVPATH:=MEAT;
                         LINK:=NIL
                   END;
        ADD(T);
        IF (T <> NIL) THEN POP[-1]:=POP[-1] + MEATPOP;

        T:=NEWPTR;
        WITH T^ DO BEGIN KIND:=LARTOPUP;
                         TIME:=DEVTIME;
                         POPULATION:=BETWPOP;
                         DEVPATH:=BETWEEN;
                         LINK:=NIL
```

```
        ADD(T);
        IF (T <> NIL) THEN POP[-1]:=POP[-1] + BETWPOP

      END {then Devpath=Hull}
    ELSE
      BEGIN {else Devpath must = Tree-mummy (ground mummy eggs die) }
        POP[0]:=POP[0] - C^.POPULATION; {decrement egg pop.}
        DEVTIME:=WHEN(DDINLARVA, {above} LARVATHR );

        IF ( (CURRENTIME<HARVESTIME) AND (DEVTIME>HARVESTIME) )
          THEN MDDT:=1.0
          ELSE MDDT:=MUMMYDROP(DEVTIME);

        TREEPOP:=((1.0 - MDDT) / (1.0 - MUMMYDROP(CURRENTIME))) *
                 C^.POPULATION ;
        GROUNDPOP:=C^.POPULATION - TREEPOP; {fall after hatching}
        TREEPOP:=TREEPOP * TMFRPUPT;
        GROUNDPOP:=GROUNDPOP * ( (GMFRPUPT + TMFRPUPT) / 2.0 ) ;
                                          { ^-- larval mortality }
        T:=NEWPTR;
        WITH T^ DO BEGIN KIND:=LARTOPUP;
                         TIME:=DEVTIME;
                         POPULATION:=TREEPOP;
                         DEVPATH:=TREEMUMMY;
                         LINK:=NIL
                   END;
        ADD(T);
        IF (T <> NIL) THEN POP[-1]:=POP[-1] + TREEPOP;

        T:=NEWPTR;
        WITH T^ DO BEGIN KIND:=LARTOPUP;
                         TIME:=DEVTIME;
                         POPULATION:=GROUNDPOP;
                         DEVPATH:=GROUNDMUMMY;
                         LINK:=NIL
                   END;
        ADD(T);
        IF (T <> NIL) THEN POP[-1]:=POP[-1] + GROUNDPOP

      END {else Devpath=Tree mummy}
END; {Hatcher}


PROCEDURE PUPATE; {Lartopup event generates Puptoadlt event}
VAR T:EVENTPTR;
    TREEPOP,GROUNDPOP:REAL;
    MDDT:REAL; {As in Hatcher}
    DEVTIME:REAL;

BEGIN
  POP[-1]:=POP[-1] - C^.POPULATION;
  CASE C^.DEVPATH OF
          MEAT : BEGIN
                   T:=NEWPTR;
                   WITH T^ DO
                     BEGIN KIND:=PUPTOADLT;
                           TIME:=WHEN(DDINPUPA, {above} PUPATHR);
                           POPULATIION:=C^.POPULATION * MTFRRMRG;
                           DEVPATH:=MEAT;              { ^ - pupal mortality}
                           LINK:=NIL
                     END;
                   ADD(T);
                   IF (T <> NIL) THEN POP[7]:=POP[7] + T^.POPULATION
                 END;
        BETWEEN : BEGIN
```

```
                    WITH T^ DO
                      BEGIN KIND:=PUPTOADLT;
                            TIME:=WHEN(DDINPUPA, {above} PUPATHR);
                            POPULATIION:=C^.POPULATION * BTFRRMRG;
                            DEVPATH:=BETWEEN;              { ^ - pupal mortality}
                            LINK:=NIL
                      END;
                    ADD(T);
                    IF (T <> NIL) THEN POP[7]:=POP[7] + T^.POPULATION
                  END;
      TREEMUMMY : BEGIN
                    DEVTIME:=WHEN(DDINPUPA, {above} PUPATHR);

                    IF ( (CURRENTIME<HARVESTIME) AND (DEVTIME>HARVESTIME) )
                      THEN MDDT:=1.0
                      ELSE MDDT:=MUMMYDROP(DEVTIME);

                    TREEPOP:=((1.0-MDDT)/(1.0-MUMMYDROP(CURRENTIME)))
                             * C^.POPULATION ;
                    GROUNDPOP:=C^.POPULATION - TREEPOP ;
                    TREEPOP:=TREEPOP * TMFRRMRG;              {pupal mortality}
                    GROUNDPOP:=GROUNDPOP * ( (GMFRRMRG/2.0) + (TMFRRMRG/2.0) );

                    T:=NEWPTR;
                    WITH T^ DO BEGIN KIND:=PUPTOADLT;
                                     TIME:=DEVTIME;
                                     POPULATION:=TREEPOP;
                                     DEVPATH:=TREEMUMMY;
                                     LINK:=NIL
                               END;
                    ADD(T);
                    IF (T <> NIL) THEN POP[7]:=POP[7] + TREEPOP;

                    T:=NEWPTR;
                    WITH T^ DO BEGIN KIND:=PUPTOADLT;
                                     TIME:=DEVTIME;
                                     POPULATION:=GROUNDPOP;
                                     DEVPATH:=GROUNDMUMMY;
                                     LINK:=NIL
                               END;
                    ADD(T);
                    IF (T <> NIL) THEN POP[7]:=POP[7] + GROUNDPOP
                  END; {Case of Devpath=Treemummy}
  GROUNDMUMMY : BEGIN
                    T:=NEWPTR;
                    WITH T^ DO BEGIN KIND:=PUPTOADLT;
                                     TIME:=WHEN(DDINPUPA, {above} PUPATHR);
                                     POPULATION:=C^.POPULATION * GMFRRMRG;
                                     DEVPATH:=GROUNDMUMMY; {pupal mort.-^ }
                                     LINK:=NIL
                               END;
                    ADD(T);
                    IF (T <> NIL) THEN POP[7]:=POP[7] + T^.POPULATION
                  END
  END {Case C^.Devpath of}

END; {Pupate}


PROCEDURE REEMERGE; {Puptoadlt event generates Lay events and Death event }
VAR DEATHTIME:REAL;
    T:EVENTPTR;

BEGIN
  POP[7]:=POP[7] - C^.POPULATION;
```

```
   IF (C^.POPULATION < MINPOP*0.1)

       THEN
          {WRITELN(' Event of kind ',ORD(C^.KIND),' discarded');}

       ELSE                {schedule egglaying and adult deaths}
         BEGIN
          POP[8]:=POP[8] + C^.POPULATION;

          DEATHTIME:=WHEN(DDINADLT, {above} ADULTTHR );


          {Egg layings: }
          LAYWHILE(C^.POPULATION / 2.0,CURRENTIME,DEATHTIME,0.0);

          {Death: }
          T:=NEWPTR;
          WITH T^ DO BEGIN KIND:=DEATH;
                           TIME:=DEATHTIME;
                           POPULATION:=C^.POPULATION;
                           DEVPATH:=NONE;
                           LINK:=NIL
                     END;
          ADD(T)
       END
END; {Re-emerge}


PROCEDURE LAYER; {Lay event generates Hatch event}
VAR DEVTIME:REAL; {Calendar time of new hatch }
    MDDT:REAL; {As in Hatcher}
    T:EVENTPTR;
    I:INTEGER;


BEGIN
  EGGTRAPCOUNT:=EGGTRAPCOUNT+C^.POPULATION;
  DEVTIME:=WHEN(DDINEGG, {above} EGGTHR);
  IF ( (C^.DEVPATH = HULL)  OR  (C^.DEVPATH = PHULL) )
    THEN
       BEGIN
          C^.POPULATION:=C^.POPULATION * HLFRHTCH;
          T:=NEWPTR;
          WITH T^ DO BEGIN KIND:=HATCH;
                           TIME:=DEVTIME;
                           POPULATION:=C^.POPULATION;
                           DEVPATH:=C^.DEVPATH;
                           LINK:=NIL
                     END;
          ADD(T)
       END
    ELSE {Devpath = TM OR PTM}
       BEGIN
          C^.POPULATION:=C^.POPULATION * TMFRHTCH;
          IF (C^.DEVPATH = TREEMUMMY)
            THEN {Cash crop}
              BEGIN
                IF ( (CURRENTIME<HARVESTIME) AND (DEVTIME>HARVESTIME) )
                  THEN MDDT:=1.0
                  ELSE MDDT:=MUMMYDROP(DEVTIME);


                C^.POPULATION:=C^.POPULATION*( (1.0-MDDT) /
                                               (1.0-MUMMYDROP(CURRENTIME))    );
```

```
                                TIME:=DEVTIME;
                                POPULATION:=C^.POPULATION;
                                DEVPATH:=TREEMUMMY;
                                LINK:=NIL
                              END;
                    ADD(T)
                  END
                ELSE {Devpath = PTM}
                  BEGIN
                    IF ( (CURRENTIME<HARVESTIME) AND (DEVTIME>HARVESTIME) )
                      THEN MDDT:=1.0
                      ELSE MDDT:=PMUMMYDROP(DEVTIME);

                    C^.POPULATION:=C^.POPULATION*( (1.0-MDDT)    /
                                                   (1.0-PMUMMYDROP(CURRENTIME))    );
                    T:=NEWPTR;
                    WITH T^ DO BEGIN KIND:=HATCH;
                                TIME:=DEVTIME;
                                POPULATION:=C^.POPULATION;
                                DEVPATH:=PTREEMUMMY;
                                LINK:=NIL
                              END;
                    ADD(T)
                  END
          END; {Else Devpath = TM OR PTM}

      IF ( T <> NIL ) THEN POP[0]:=POP[0] + C^.POPULATION

END; { L a y e r }


PROCEDURE FUNERAL; {Death event only affects global population count, Pop[8] }
BEGIN
  POP[8]:=POP[8] - C^.POPULATION
END; {Funeral}


PROCEDURE SPRAYER;
VAR S:EVENTPTR;
    I:INTEGER;
    LTOPSEEN,LAYSEEN,DEATHSEEN:INTEGER;

BEGIN
  IF ( C^.DEVPATH = LARVACIDAL )
    THEN
      BEGIN
        {Global effect (immediate) : }
        FOR I:=1 TO 6 DO POP[I]:=POP[I] * ( 1.0 - IEFFLONL );
        POP[-1]:=POP[-1] * ( 1.0 - IEFFLONL );

        {Effect within the list (future) : }
        LTOPSEEN:=0;
        S:=C^.LINK;
        WHILE (    ((S^.TIME-CURRENTIME) < DURLARSP) OR
                   (LTOPSEEN < COUNT[LARTOPUP]  )       )
              AND (S^.KIND <> FINAL)
        DO BEGIN
             IF (S^.KIND=LARTOPUP)
               THEN
                 BEGIN
                   LTOPSEEN:=LTOPSEEN + 1;
                   S^.POPULATION:=S^.POPULATION * (1.0 - IEFFLONL)
                 END
               ELSE
```

```
                    IF ((S^.KIND=MATCH) AND (S^.TIME-CURRENTIME) < DURLARSP)
                      THEN S^.POPULATION:=S^.POPULATION *
                                          (1.0-(IEFFLONL*(1.0-((S^.TIME-CURRENTIME)/
                                                              DURLARSP))))
                  END;
              S:=S^.LINK
            END {While loop}
      END {Then Devpath=Larvacidal}
    ELSE
      BEGIN {else Devpath=Adulticidal}
        {Global effect (immediate) : }
        POP[8] := POP[8] * (1.0 - IEFFAONA) ;

        {Effect within list (future) : }
        LAYSEEN:=0;
        DEATHSEEN:=0;
        S:=C^.LINK;
        WHILE ( (((S^.TIME - CURRENTIME) < DURADLSP) OR
                 (LAYSEEN < COUNT[LAY]           ) OR
                 (DEATHSEEN < COUNT[DEATH]       )   )
                AND (S^.KIND <> FINAL)                   )  DO
          BEGIN
            IF (S^.KIND = LAY)
               THEN
                 BEGIN LAYSEEN:=LAYSEEN + 1 ;
                       S^.POPULATION:=S^.POPULATION * (1.0 - IEFFAONA)
                 END
               ELSE
                 BEGIN
                   IF (S^.KIND = DEATH)
                     THEN
                       BEGIN DEATHSEEN:=DEATHSEEN + 1 ;
                             S^.POPULATION:=S^.POPULATION * (1.0-IEFFAONA)
                       END
                     ELSE
                       BEGIN
                         IF ( (S^.KIND = PUPTOADLT) AND
                              ((S^.TIME-CURRENTIME) < DURADLSP) )

                           THEN S^.POPULATION:=S^.POPULATION * ( 1.0-(IEFFAONA *
                                                ( 1.0 - ( (S^.TIME-CURRENTIME) /
                                                  DURADLSP ) )) )
                       END
                 END;
            S:=S^.LINK
          END {While loop}

      END {else Devpath=Adulticidal}
END; {Sprayer}

PROCEDURE KNOCKER;
BEGIN

END; {Knocker}

PROCEDURE SWEEPER;
BEGIN

  D; {Sweeper}


PROCEDURE OUTPUTER;
VAR T:EVENTPTR;
    POINT:ARRAY[1..4] OF INTEGER;
    PRINTOUT:ARRAY[1..4] OF CHAR;
```

```
    TC: CHAR;

BEGIN
  IF HARDCOPY THEN
    BEGIN
      IF (C^.TIME >= STARTIME) THEN
        BEGIN
          WRITE(GRAPHILE, CURRENTIME:7:2, ' ');

          FOR I:= 1 TO 4 DO PRINTOUT[I]:=SYM[I];

          POINT[1]:=ROUND( POP[0] * SFEGGS );
          POINT[2]:=ROUND( POP[1] * SFLARVA1 );
          POINT[3]:=ROUND( POP[-1] * SFTOTLARVA );
          POINT[4]:=ROUND( POP[8] * SFADULTS );


          FOR I:= 1 TO 4 DO IF POINT[I] >  70 THEN POINT[I]:=70;


          {SORT: }


          FOR I:= 1 TO 3 DO
            FOR J:= I TO 4 DO
              BEGIN
                IF ( (I<>J) AND (POINT[I] > POINT[J])  )
                THEN
                  BEGIN
                    TEMP:=POINT[I];
                    POINT[I]:=POINT[J];
                    POINT[J]:=TEMP;

                    TC:=PRINTOUT[I];
                    PRINTOUT[I]:=PRINTOUT[J];
                    PRINTOUT[J]:=TC;
                  END;
              END;



          LASTPOINT:=0;

          FOR J:= 1 TO 3 DO
            BEGIN
              IF PRINTOUT[J] <> ' ' THEN
              IF ( (POINT[J] <> POINT[J+1])  OR (PRINTOUT[J+1]=' ')  )
              THEN
                BEGIN
                  FOR I:=LASTPOINT TO (POINT[J]-1) DO
                    WRITE(GRAPHILE, ' ');
                    WRITE(GRAPHILE, PRINTOUT[J]);
                    LASTPOINT:=I+1;
              END;
            END;

          IF PRINTOUT[4] <> ' '
          THEN
            BEGIN
              FOR I:=LASTPOINT TO (POINT[4]-1) DO
                WRITE(GRAPHILE, ' ');
              WRITE(GRAPHILE, PRINTOUT[4]);
            END;
          WRITELN(GRAPHILE)
  END;
```

```
      T:=NEWPTR;
      WITH T^ DO BEGIN
                   KIND:=PAUSE;
                   TIME:=CURRENTIME + NEXTOUT;
                   POPULATION:=MINPOP;
                   DEVPATH:=NONE;
                   LINK:=NIL
                 END;
      ADD(T)
    END
END;

PROCEDURE EGGPRINTER;

  VAR SCALEDCOUNT, I: INTEGER;

  BEGIN
    IF FIRSTRAP
    THEN BEGIN
           FIRSTRAP:=FALSE;
           EGGTRAPCOUNT:=0.0;
           LASTPRINT:=TRUNC(CURRENTIME);
         END
    ELSE BEGIN
           FOR I:=(LASTPRINT+1) TO (TRUNC(CURRENTIME)-1) DO
             WRITELN(GRAPHILE, I:4, '.00');
           WRITE(GRAPHILE, CURRENTIME:7:2);
           SCALEDCOUNT:=ROUND(EGGTRAPCOUNT*0.065);
           IF (SCALEDCOUNT>70) THEN SCALEDCOUNT:=70;
           FOR I:=0 TO (SCALEDCOUNT-1) DO
             WRITE(GRAPHILE, ' ');
           WRITELN(GRAPHILE, 'e');
           EGGTRAPCOUNT:=0.0;
           LASTPRINT:=TRUNC(CURRENTIME);
         END;
  END;


PROCEDURE DUMPLIST;
VAR E:EVENTPTR;

BEGIN
  IF DUMP
    THEN
      BEGIN
          E:=CURRENTEVENT;
          WHILE E <> NIL DO
            BEGIN
              WRITE('Ord(kind)=', ORD(E^.KIND):2, 'Pop=', E^.POPULATION:8:1);
              WRITELN(' Ord(path)=', ORD(E^.DEVPATH):2);
              E:=E^.LINK
            END
      END
END;

BEGIN { O n e _ e v e n t:    ("Central" simulation procedure) }
  WRITELN('ONE_EVENT.                                                    ');

  C:=CURRENTEVENT; {Local copy to speed up "global" reference}
  IF ( C^.TIME > STOPTIME )
    THEN
      CONT:=FALSE
    ELSE {continue simulation}
```

```
        CURRENTIME:=C^.TIME;

        DUMPLIST;

        CASE  (C^.KIND)  OF
              HATCH : HATCHER;     {unhatched egg --> larva}
           LARTOPUP : PUPATE;      {larva         --> pupa }
          PUPTOADLT : REEMERGE;    {pupa          --> adult}
                LAY : LAYER;       {adult (i.e. emerged pupa) deposits eggs}
              DEATH : FUNERAL;     {adult (i.e. emerged pupa) dies          }
              SPRAY : SPRAYER;     {kills present and emerging larva or adults}
              KNOCK : KNOCKER;     {shakes mummies off of trees onto ground}
              SWEEP : SWEEPER;     {removes mummies from ground             }
              PAUSE : OUTPUTER;    {outputs statistics to somewhere}
         EGGSAMPLE : EGGPRINTER;   {outputs eggtrap sampling}
              FINAL : CONT:=FALSE   (* GOODBYE   {Exits program} *)
        END; {Case Kind of}

      END {else continue simulation}

END; {of O n e _ e v e n t}
```

```
SEGMENT PROCEDURE NEWDATA;

{This is an include file for N.O.W. as of Aug-28-79.}

VAR NEEDMORE: BOOLEAN;
    NEWCOMMAND: CHAR;


PROCEDURE MORETHRESHOLDS;

  VAR CH: CHAR;

  BEGIN
    WRITE(CHR(13), 'Change Larval threshold ? ');
    READ(CH);
    IF CH IN['Y', 'y'] THEN
      BEGIN
        WRITE(CHR(13), 'Enter new threshold : ');
        READLN(LARVATHR);
      END
    ELSE WRITELN;
    WRITE(CHR(13), 'Change Pupa threshold ? ');
    READ(CH);
    IF CH IN['Y', 'y'] THEN
      BEGIN
        WRITE(CHR(13), 'Enter new threshold : ');
        READLN(PUPATHR);
      END
    ELSE WRITELN;
    WRITE(CHR(13), 'Change Adult threshold ? ');
    READ(CH);
    IF CH IN['Y', 'y'] THEN
      BEGIN
        WRITE(CHR(13), 'Enter new threshold : ');
        READLN(ADULTTHR);
      END
    ELSE WRITELN;
    WRITE(CHR(13), 'Change calling threshold ? ');
    READ(CH);
    IF CH IN['Y', 'y'] THEN
      BEGIN
        WRITE(CHR(13), 'Enter new threshold : ');
        READLN(CALLTHR);
      END
    ELSE WRITELN;
  END;


PROCEDURE NEWTHRESHOLDS;

  VAR CH: CHAR;
```

```
    WRITELN(CHR(12), 'The present temperature thresholds are : ');
    WRITELN(CHR(13),'    Eggs       ',EGGTHR:3:1);
    WRITELN('    Larva      ',LARVATHR:3:1);
    WRITELN('    Pupa       ',PUPATHR:3:1);
    WRITELN('    Adult      ',ADULTTHR:3:1);
    WRITELN('    Calling    ',CALLTHR:3:1,CHR(13));
    WRITE('Do you want to change a threshold ? ');
    READ(CH);
    IF CH IN['Y','y'] THEN
      BEGIN
        WRITE(CHR(13),CHR(13),'Change egg threshold ? ');
       READ(CH);
       IF CH IN['Y','y'] THEN
         BEGIN
           WRITE(CHR(13),'Enter new threshold : ');
           READLN(EGGTHR);
         END
       ELSE WRITELN;
       MORETHRESHOLDS;
      END;
    WRITE(CHR(12));
    GOTOXY(0,4);
    WRITELN('The present minimum population is  ',MINPOP:1:2);
    WRITE(CHR(13),'Do you want to change the minimum population  ? ');
    READ(CH);
    IF CH IN['Y','y'] THEN
      BEGIN
        WRITE(CHR(13),CHR(13),'Enter new minimun population : ');
        READLN(MINPOP);
      END;
  END;      {  end of NEWTHRESHOLDS  }

PROCEDURE NEWTIMES;
VAR CH:CHAR;

BEGIN
  WRITELN( CHR(12) , 'Current Start Time for simulation is ',STARTIME:7:2);
  WRITELN('Current Stop Time for simulation is ',STOPTIME:7:2);
  REPEAT
    WRITELN;
    IF ( NOT (HASBEGUN) )
      THEN
        BEGIN
          WRITE('Change Start Time? ');
          READ(CH);
          WRITELN;
          IF (CH IN ['Y','y'] )
            THEN BEGIN WRITE('Enter new Start Time--> ');
                       READLN(STARTIME);
                       CURRENTIME:=STARTIME;
                       WRITELN
                 END
        END;

    IF ( (STARTIME < STOPTIME)  AND  (STOPTIME <= MAXTIME) )
      THEN BEGIN WRITE('Change Stop Time? ');
                 READ(CH);
                 WRITELN
           END
      ELSE CH:='Y' ;
    IF ( CH IN ['Y','y'] )
      THEN BEGIN
             WRITE('Enter new Stop Time--> ');
             READLN(STOPTIME);
             WRITELN
```

```
    WRITELN;
    IF (STARTIME >= STOPTIME)
      THEN WRITELN(CHR(7), 'Start must precede Stop!' );
    IF (STARTIME < 0.0)
      THEN WRITELN(CHR(7),'Min. time is  0.00');
    IF (STOPTIME > MAXTIME)
      THEN  WRITELN(CHR(7),'Max. time is ',MAXTIME:7:2);

    OLDTIME:=STARTIME;

    WRITELN;
    WRITELN('Start Time for simulation is ',STARTIME:7:2);
    WRITELN('Stop Time for simulation is ',STOPTIME:7:2);
    WRITE(CHR(13),'Type <cr> to continue  ');
    READLN

  UNTIL ( (STARTIME < STOPTIME) AND (STOPTIME <= MAXTIME) );
  IF (NOT (HASBEGUN)) THEN NEWTHRESHOLDS;
END; {Newtimes}


PROCEDURE INITPOP;

CONST

PROFILE='Eggs |      L a r v a   ( p e r   I n s t a r )      | Pupa | Adults';
INSTARS='  -  |    1       2       3       4       5       6  |   -  |   -  ';

VAR I:INTEGER;
    K:EVENTKIND;
    E:EVENTPTR;
    CH:CHAR;
    INSECTS:SET OF EVENTKIND;
    NEWPOP:ARRAY[-1..8] OF REAL;
    NEWCOUNT:ARRAY[HATCH .. DEATH] OF INTEGER;

PROCEDURE EVENMOREHELP;

  VAR  INFILE:TEXT;
       LINEDATA:STRING;

  BEGIN
     IF TESTFILE('INFO.TEXT')
     THEN
       BEGIN
         RESET(INFILE,'INFO.TEXT');
         WRITE(CHR(11));
         WHILE NOT EOF(INFILE) DO
            BEGIN
              READLN(INFILE,LINEDATA);
              WRITELN(LINEDATA);
            END;
        END
     ELSE WRITE(CHR(7),'Information file is not on disk !');
     CLOSE(INFILE);
 END;


PROCEDURE DISPLAY;
BEGIN
  WRITELN(CHR(12),CHR(13) );
  WRITELN('Population profile at ' , CURRENTIME:7:2 , ' model time ' ,
          ' (Weather file is ' , TEMPNAME , ') :' );
```

```
  WRITELN('Popula-     ',INSTARS);
  WRITE(  'tion  ->',POP[0]:8:1);
  FOR I:=1 TO 8 DO WRITE(POP[I]:8:1);
  WRITELN(CHR(13),CHR(13),'No. of');
  WRITE('cohorts -> ',COUNT[HATCH]:3,'    ');
  FOR I:=1 TO 6 DO IF (POP[I] > 0.0)
                   THEN WRITE(1:5,'    ')
                   ELSE WRITE(0:5,'    ');
  WRITE( COUNT[PUPTOADLT]:5,'    ');
  WRITELN(COUNT[DEATH]:5);
  FOR I:=1 TO 80 DO WRITE('_');
  WRITELN
END; {Display (current profile) }

PROCEDURE CLEAR;
BEGIN
  WRITELN('Please wait while I clear the current population.');

  FOR K:=HATCH TO DEATH DO COUNT[K]:=0 ;
  FOR I:=-1 TO 8 DO POP[I]:=0.0 ;
  LASTUPDATE:=CURRENTIME;

  INSECTS:=[HATCH,LARTOPUP,PUPTOADLT,LAY,DEATH];

  WHILE (CURRENTEVENT^.KIND  IN  INSECTS ) DO {Delete insects at head of list}
    BEGIN
      TEMPFREE:=FREE;
      FREE:=CURRENTEVENT;
      CURRENTEVENT:=CURRENTEVENT^.LINK;
      FREE^.LINK:=TEMPFREE
    END;

  IF (CURRENTEVENT^.KIND <> FINAL) THEN
    BEGIN
      E:=CURRENTEVENT;
      WHILE (E^.LINK^.KIND <> FINAL)
        DO IF (E^.LINK^.KIND  IN  INSECTS)
             THEN BEGIN
                    TEMPFREE:=FREE;
                    FREE:=E^.LINK;
                    E^.LINK:=FREE^.LINK;
                    FREE^.LINK:=TEMPFREE
                  END
             ELSE E:=E^.LINK
    END
END; {Clear}

PROCEDURE REPLAY; {i.e. "RE-disPLAY" a blank profile}
BEGIN
  GOTOXY(0,11);
  WRITELN('New Profile for ',CURRENTIME:7:2,' model time :',CHR(29) );
  WRITELN(PROFILE:80);
  WRITELN('Popula-     ',INSTARS);
  WRITE(  'tion  ->     ?');
  FOR I:=1 TO 8 DO WRITE('       ?');
  WRITELN(CHR(13) );
  WRITELN('No. of');
  WRITE('cohorts ->   ?');
  FOR I:=1 TO 6 DO WRITE('       1');
  WRITELN('       ?       ?')
END; {Replay}

PROCEDURE GETPUPA;
BEGIN
  REPEAT
```

```
    GOTOXY(0,19);
    WRITE('How many pupa, total? ',CHR(29) );
    READLN(NEWPOP[7])
  UNTIL (NEWPOP[7] >= 0.0);
  GOTOXY(64,14);
  WRITE(NEWPOP[7]:8:1);

  IF (NEWPOP[7] > 0.0)
    THEN REPEAT
           GOTOXY(69,18);WRITE('^');
           GOTOXY(0,20);
           WRITE('Divided into how many cohorts? ',CHR(29) );
           READLN(NEWCOUNT[PUPTOADLT])
         UNTIL (NEWCOUNT[PUPTOADLT] > 0)
    ELSE NEWCOUNT[PUPTOADLT]:=0;
  GOTOXY(67,17);
  WRITE(NEWCOUNT[PUPTOADLT]:3);
  GOTOXY(69,18);WRITE(' ')
END; {Getpupa}

PROCEDURE GETADULTS;
BEGIN
  REPEAT
     GOTOXY(69,15);WRITE('         ^');
     GOTOXY(0,19);
     WRITE('How many adults, total? ',CHR(29) );
     READLN(NEWPOP[8])
  UNTIL (NEWPOP[8] >= 0.0);
  GOTOXY(72,14);
  WRITE(NEWPOP[8]:8:1);

  IF (NEWPOP[8] > 0.0)
    THEN REPEAT
           GOTOXY(77,18);WRITE('^');
           GOTOXY(0,20);
           WRITE('Divided into how many cohorts? ',CHR(29) );
           READLN(NEWCOUNT[DEATH])
         UNTIL (NEWCOUNT[DEATH] > 0)
    ELSE NEWCOUNT[DEATH]:=0;
  GOTOXY(75,17);
  WRITE(NEWCOUNT[DEATH]:3);
  GOTOXY(77,15);WRITE(' ');
  GOTOXY(77,18);WRITE(' ')
END; {Getadults}

PROCEDURE SCHEDULE;
VAR I:INTEGER;
    REM:REAL; {REMaining deg.-days }
    T:EVENTPTR;

PROCEDURE PUTEGGS;
VAR MAINPATH:PATH;

BEGIN
  IF ((HULLCRACK(CURRENTIME)*NUTSPT) >= ((1-MUMMYDROP(CURRENTIME))*IMUMMYPT))
    THEN MAINPATH:=HULL
    ELSE MAINPATH:=TREEMUMMY;

  FOR I:=NEWCOUNT[HATCH] DOWNTO 1 DO
    BEGIN
      REM:=(I * DDINEGG) / NEWCOUNT[HATCH];
      T:=NEWPTR;
      WITH T^ DO BEGIN TIME:=WHEN(REM, {above} EGGTHR);
                       KIND:=HATCH;
                       POPULATION:=NEWPOP[0] / NEWCOUNT[HATCH];
```

```
                                LINK:=NIL
                        END;
        ADD(T)
      END
END; {Puteggs}

PROCEDURE PUTLARVA;
VAR MAINPATH:PATH;

BEGIN
  IF (    (HULLCRACK(CURRENTIME) * NUTSPT)
                              >=
        ((1-MUMMYDROP(CURRENTIME)) * IMUMMYPT)   )
     THEN MAINPATH:=MEAT
     ELSE MAINPATH:=TREEMUMMY;

  FOR I:=1 TO 6
    DO IF (NEWPOP[I] > MINPOP)
         THEN BEGIN REM:=MINREMIN[I-1]; {i.e. "Maxremin[i]" = Minremin[i-1] }
                    T:=NEWPTR;
                    WITH T^ DO
                      BEGIN TIME:=WHEN(REM, {above} LARVATHR);
                            KIND:=LARTOPUP;
                            POPULATION:=NEWPOP[I];
                            DEVPATH:=MAINPATH;
                            LINK:=NIL
                      END;
                    ADD(T)
              END
END; {Putlarva}

PROCEDURE PUTPUPA;
BEGIN
  FOR I:=NEWCOUNT[PUPTOADLT] DOWNTO 1 DO
    BEGIN
      REM:=(I * DDINPUPA) / NEWCOUNT[PUPTOADLT];
      T:=NEWPTR;
      WITH T^ DO BEGIN TIME:=WHEN(REM, {above} PUPATHR);
                       KIND:=PUPTOADLT;
                       POPULATION:=NEWPOP[7] / NEWCOUNT[PUPTOADLT];
                       DEVPATH:=MEAT; {w.l.o.g.}
                       LINK:=NIL
                 END;
      ADD(T)
    END
END; {Putpupa}

PROCEDURE PUTADULTS;
VAR FEMALES,
    DEATHTIME:REAL;

BEGIN
  FEMALES:=(NEWPOP[8] / NEWCOUNT[DEATH])  /  2.0 ;

  FOR I:=NEWCOUNT[DEATH] DOWNTO 1 DO
    BEGIN
      REM:=(I * DDINADLT) / NEWCOUNT[DEATH];
      DEATHTIME:=WHEN(REM, {above} ADULTTHR);

      {The Ith. cohort's (remaining) Egg Layings: }
      LAYWHILE(FEMALES,CURRENTIME,DEATHTIME,(DDINADLT-REM));

      {The Ith. cohort's Death event: }
      T:=NEWPTR;
      WITH T^ DO BEGIN KIND:=DEATH;
```

```
                        POPULATION:=NEWPOP[8] / NEWCOUNT[DEATH];
                        DEVPATH:=NONE;
                        LINK:=NIL
                    END;
        ADD(T)
      END
  END; {Putadults}

  BEGIN {Schedule}
    GOTOXY(0,22);
    WRITELN('Please wait while I digest this information.');

    IF ( NEWCOUNT[HATCH] > 0 )  THEN PUTEGGS;

    PUTLARVA;

    IF ( NEWCOUNT[PUPTOADLT] > 0 )  THEN PUTPUPA;
    IF ( NEWCOUNT[DEATH] > 0 )  THEN PUTADULTS;

    {Now transfer Newpop[] (local)  --->  Pop[] (global) :  }
    FOR I:=0 TO 8 DO POP[I]:=NEWPOP[I];
    FOR I:=1 TO 6 DO POP[-1]:=POP[-1] + POP[I]

  END; {Schedule}


  BEGIN { I n i t _ p o p }

    WRITE(CHR(12) );

    WRITE('Need an explanation? ');
    READ(CH);
    WRITELN;
    IF ( CH  IN  ['Y','y'] ) THEN
      BEGIN
        EVENMOREHELP;
        WRITELN;
        WRITE('   Type <CR> when you are ready to see the current profile.');

        READLN
      END;

    WRITE( CHR(12) , 'Please wait while I sort out the larval instars.');
    UPDATE;

    DISPLAY;

    WRITE('K(eep  or  R(eplace  this profile? ');
    READ(CH);
    WRITELN;
    IF (CH IN ['R','r'] ) THEN
      BEGIN
        HASBEGUN:=TRUE;

        FOR I:=-1 TO 8 DO NEWPOP[I]:=0.0;
        FOR K:=HATCH TO DEATH DO NEWCOUNT[K]:=0;

        CLEAR;
        REPLAY;

        REPEAT
          GOTOXY(13,15); WRITE('^');
          GOTOXY(0,19);
          WRITE('How many eggs, total? ',CHR(29) );
          READLN(NEWPOP[0])
```

```
      GOTOXY(8,14);
      WRITE(NEWPOP[0]:8:1);

      IF (NEWPOP[0] > 0.0)
        THEN BEGIN
               REPEAT
                 GOTOXY(13,18);WRITE('^');
                 GOTOXY(0,20);
                 WRITE('Divided into how many cohorts? ',CHR(29) );
                 READLN(NEWCOUNT[HATCH] )
               UNTIL (NEWCOUNT[HATCH] > 0);
               GOTOXY(0,20);
               WRITE(CHR(29) )
             END
        ELSE NEWCOUNT[HATCH]:=0;
      GOTOXY(11,17);
      WRITE(NEWCOUNT[HATCH]:3);
      GOTOXY(13,18);WRITE(' ');

      FOR I:=1 TO 6 DO
        BEGIN
          REPEAT
            GOTOXY(5+(8*I),15);
            WRITE('         ^');
            GOTOXY(0,19);
            WRITE('How many larva in instar # ',I,'? ',CHR(29) );
            READLN(NEWPOP[I])
          UNTIL (NEWPOP[I] >= 0.0);
          GOTOXY(8+(8*I),14);
          WRITE(NEWPOP[I]:8:1);
          NEWPOP[-1]:=NEWPOP[-1] + NEWPOP[I];
          IF (NEWPOP[I] > 0.0) THEN NEWCOUNT[LARTOPUP]:=NEWCOUNT[LARTOPUP] + 1
        END;

      GETPUPA;

      GETADULTS;

      SCHEDULE

    END {If Replace current population}

END; {I n i t _ p o p}


PROCEDURE WEATHER;
VAR GOTFILE:BOOLEAN;
    CH:CHAR;
    NEWFILE:STRING;
BEGIN
  WRITE(CHR(12) );
  IF (NODEFAULT)
    THEN WRITELN('The default file, ',DEFAULT,', was not available.')
    ELSE WRITELN('The default file is ',DEFAULT);
  WRITELN('The current temperature file is ',TEMPNAME);
  GOTFILE:=FALSE;
  REPEAT
    WRITELN;
    WRITE('Do you want the:    D(efault,  C(urrent,  or  A(nother  file?');
    REPEAT
      READ(KEYBOARD,CH)
    UNTIL ( CH IN ['D','d','C','c','A','a'] );
    WRITELN;
    CASE CH OF
      'D','d':BEGIN {Try for default}
```

```
                    THEN BEGIN
                           WRITELN('Default kept.');
                           GOTFILE:=TRUE
                         END
                    ELSE BEGIN
                           NODEFAULT:=NOT ( SENSEFILE(DEFAULT) );
                           IF NODEFAULT
                             THEN WRITELN(DEFAULT,' is not available; ',
                                          ' current temperature file is still ',
                                          TEMPNAME)
                             ELSE BEGIN
                                    TEMPNAME:=DEFAULT;
                                    RESET(TEMPS,CONCAT(TEMPNAME,'.TEMP') );
                                    READFILE;
                                    CLOSE(TEMPS);
                                    WRITELN('Temperature file is ',TEMPNAME);
                                    GOTFILE:=TRUE
                                  END
                         END {of else Tempname <> Default}
              END; {case of Ch="d"}

      'C','c':BEGIN {keep current file}
                WRITELN(TEMPNAME,' kept.');
                GOTFILE:=TRUE
              END;

      'A','a':BEGIN {try for a new file}
                WRITE('Enter name of new temperature file--> ');
                READLN(NEWFILE);
                IF ( SENSEFILE( NEWFILE )  )
                  THEN BEGIN
                         TEMPNAME:=NEWFILE;
                         RESET(TEMPS,CONCAT(TEMPNAME,'.TEMP') );
                         READFILE;
                         CLOSE(TEMPS);
                         WRITELN('Temperature file is ',TEMPNAME);
                         GOTFILE:=TRUE
                       END
                  ELSE WRITELN(NEWFILE,' is not available; current ',
                                       'temperature file is still ',
                                       TEMPNAME)
              END {case of Ch="a"}
    END {Case Ch of}

  UNTIL GOTFILE
END; {W e a t h e r}

PROCEDURE SETSPRAY;
VAR P:EVENTPTR;
    CH:CHAR;
    SPRAYTIME:REAL;

BEGIN
  WRITE(CHR(12) );
  WRITELN('Pending Spray Schedule:',CHR(13) );
  IF (COUNT[SPRAY] > 0)
    THEN
      BEGIN
        WRITELN('   Time :   Type',CHR(13) );

        P:=CURRENTEVENT;
        WHILE ( P <> NIL ) DO {Find 'em.}
          BEGIN
            IF ( P^.KIND = SPRAY )
              THEN
```