CS5250 | Programming assignment 1: Diagramless crossword puzzles | Spring 1994

Due: Friday, March 11, 1994

1 Introduction

In this assignment you are asked to write a program that solves diagramless crossword puzzles. The program is to be written in Modula-3. Please write the program in one file named `CrossWord.m3`.

Write the program as clearly as possible, including reasonable comments. You may work alone or with one other student in the class. (In the other project you may not work with the same person.) Be sure to include your name and your partner's name in the comments when you turn in the project, so we will know who should get credit for the work. Please mail me your file (just one copy per group) by midnight of the due date. Use the following command on `jove.acs.unt.edu`.

```
uuencode CrossWord.m3 < CrossWord.m3 | mail ryan@csci.unt.edu
```

2 Modula-3

The goal of Modula-3 is to be as simple and safe as it can be while meeting the needs of modern systems programmers. Modula-3 descends from Mesa, Modula-2, Cedar, and Modula-2+. It also resembles its cousins Object Pascal, Oberon, and Euclid. Valid Modula-2 programs are not valid Modula-3 programs.

Modula-3 retains one of Modula-2's most successful features, the provision for explicit interfaces between modules. It adds objects and classes, exception handling, garbage collection, lightweight processes (or threads), and the isolation of unsafe features.

There is an overview article in *Byte* magazine:

> Samuel P. Harbison. "Modula-3." *Byte*, volume 15, number 12, October 1990, page 385.

The definition of Modula-3 is contained in:

> Nelson, Greg, editor. *Systems Programming with Modula-3.* Prentice Hall series in innovative technology. Prentice Hall, Englewood Cliffs, New Jersey, 1991.

For a book-length introduction to the language see:

> Samuel P. Harbison. *Modula-3.* Prentice-Hall, Englewood Cliffs, 1992.

The Modula-3 compiler is invoked by the command `m3` (if `/usr/local/bin` is on your path). For example, if the file `Hello.m3` contains the following Modula-3 program:

```
(*  A small Modula-3 program.  *)
MODULE Hello      (*  Keywords must be in ALL CAPS.          *)
EXPORTS Main;     (*  The main module must export "Main".    *)

IMPORT Wr, Stdio; (*  Obtain some standard library modules.   *)
BEGIN
    Wr.PutText (Stdio.stdout, "Hello world!\n");
END Main.
```

Then the command

```
jove> m3 Hello.m3
```

will compile it, and the executable file will be called `a.out`. The `-make` option avoids recompilation whenever possible and so it does not hurt to use that option all the time.

```
jove> m3 -make Hello.m3 -o hello
```

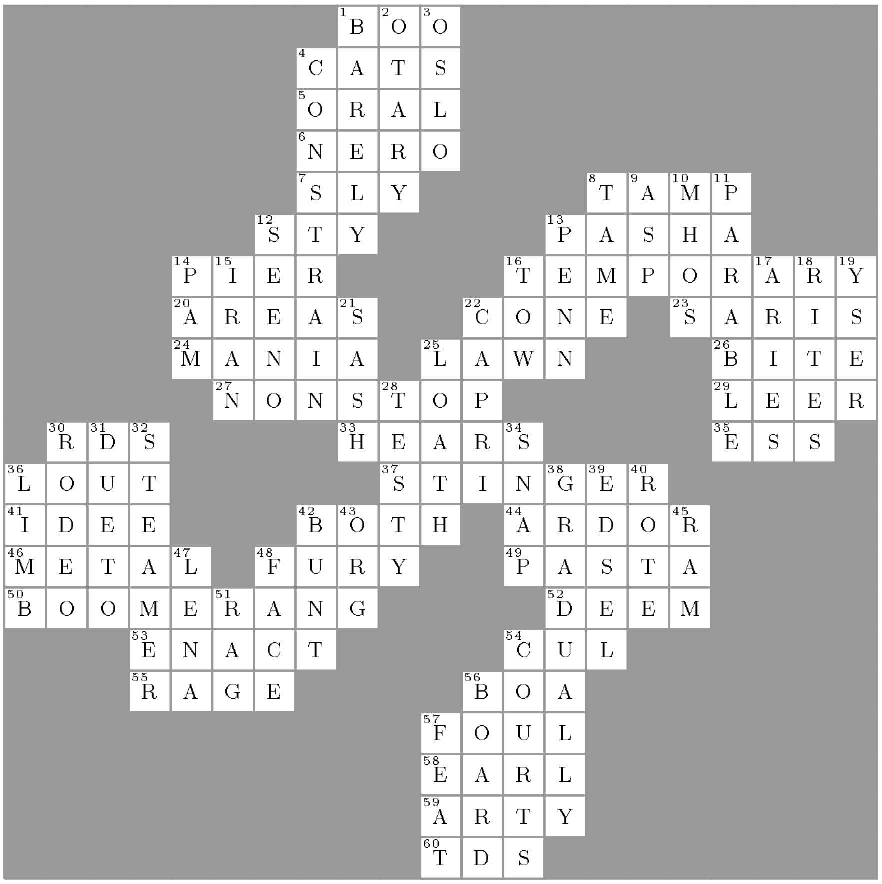
The `-o` option names the executable file to something other than `a.out`. (In the example above, the executable file is named `hello`.) An executable file is invoked by typing its name:

```
jove> hello
Hello world!
jove>
```

More information about the Modula-3 compiler can be obtained on the Solbourne by typing `man m3`.

Here is another short Modula-3 program

```
MODULE Main;
IMPORT Wr, Fmt, Stdio; (*  Obtain some standard library modules.  *)
IMPORT Text;           (*  Obtain standard string module          *)
CONST
   A = ARRAY [1..6] OF Text.T {
      "FORTRAN", "ALGOL", "LISP", "COBOL", "APL", "SNOBOL"
   };
VAR
   sum := 0;
BEGIN
    FOR i := FIRST(A) TO LAST(A) DO
        sum := sum + Text.Length (A[i]);
    END;
    Wr.PutText (Stdio.stdout, "The total length is " & Fmt.Int (sum) & ".\n");
END Main.
```



3 Diagramless crossword puzzles

A crossword puzzle is a word game in which you are given clues and you fill in a numbered grid with the answers. Some of the words run across the grid horizontally and some of the words run down vertically. Every letter of the grid is part of two words, one running across and running down. Not all the grid is filled in. Every word begins and ends adjacent to the edge or to an unused position of the diagram. Unused positions in the diagram are blackened. Usually the grid is square and often the unused portion of the grid forms a symmetric pattern.

A diagramless crossword puzzle is an especially difficult variant in which the diagram is withheld. Not only must the puzzler come up with the words, but also with the grid. Here we give an example puzzle from the Saturday *Dallas Morning News* with the solution.

ACROSS:

1 Scary yell (BOO)
4 Purring pets (CATS)
5 Spoken (ORAL)
6 Despot with a fiddle? (NERO)
7 Cunning (SLY)
8 Pack down (TAMP)
12 Place for pigs (STY)
13 High-ranking turk (PASHA)
14 Docking place (PIER)
16 Impermanent (TEMPORARY)
20 Plane surfaces (AREAS)
22 Volcano's apex (CONE)
23 Indian garments (SARIS)

24 Wild enthusiam (MANIA)
25 Gardener's concern (LAWN)
26 Nibble (BITE)
27 Like some flights (NONSTOP)
29 Lecherous look (LEER)
30 Sts. (RDS)
33 Learns (HEARS)
35 Curved letter (ESS)
36 Low fellow (LOUT)
37 Bee's weapon (STINGER)
41 — fixe (IDEE)
42 This one and that one (BOTH)
44 Warmth of feeling (ARDOR)
46 Iron, e.g. (METAL)
48 "The Sound and the —" (FURY)
49 Spaghetti (PASTA)
50 Curved wood that flies back (BOOMERANG)
52 Judge (DEEM)
53 Make into law (ENACT)
54 — -de-sac (CUL)
55 Latest fashion (RAGE)
56 Feather scarf (BOA)
57 Disgusting (FOUL)
58 Nobleman (EARL)
59 Dilettantish (ARTY)
60 Football scores: abbr. (TDS)

DOWN:

1 Scarcely (BARELY)
2 Eared seal (OTARY)
3 City in Scandinavia (OSLO)
4 Limit (CONSTRAIN)
8 Like circus animals (TAME)
9 Cleo's undoing (ASP)
10 Electrical units (MHOS)
11 Allegorical story (PARABLE)
12 "—evil" (SEENO)
13 Sean or William (PENN)
14 Shriver of tennis (PAM)
15 Abadan's land (IRAN)
16 Pull (TOW)
17 First sign of the Zodiac (ARIES)
18 Ceremonial acts (RITES)
19 River to the North Sea (YSER)
21 Cummerbund (SASH)
22 Island in the Bay of Naples (CAPRI)
25 Very unwilling (LOATH)
28 Peevish (TESTY)
30 Cattle roundup (RODEO)
31 Because of (DUETO)
32 Ship (STEAMER)
34 Cookie (SNAP)
36 Tree branch (LIMB)
38 Little by little (GRADUALLY)
39 A Ford (EDSEL)
40 Memory (ROTE)
42 Tap a baseball (BUNT)
43 Assn. (ORG)
45 Strike with great force (RAM)
47 Horne of song (LENA)
48 Confront (FACE)
51 Cleaning cloth (RAG)
54 Woos (COURTS)
56 Plank (BOARD)
57 Great deed (FEAT)

4 The project

In the project you are asked to solve diagramless crossword puzzles given the solution words in the order that they appear in the puzzle. Your program must find the diagram, i.e., where to place the words so that a proper crossword puzzle is formed. The result will be a filled-in grid of characters (. representing a black square). You will be given the interface file `CrossWord.i3`:

```
INTERFACE CrossWord;

TYPE
    Grid = REF ARRAY OF ARRAY OF CHAR;

CONST
    blk = '.';  (* black square *)

EXCEPTION no_solution;

(*
   The procedure "Solve" fills the two-dimensional grid "g" with all
   of the words given in the "across" array and all of the words in
   "down" array.  The "across" words appear horizontally, the "down"
   words appear vertically in the grid.  The resulting grid is a valid
   solution to the crossword puzzle.  We assume the list of words are
   given in the order they appear in the puzzle.  If no solution can
   be found, then the exception "no_solution" is raised.

   The pointer to the grid "g" is READONLY and cannot be changed.  We
   assume that the two-dimensional grid has been allocated and has the
   appropriate size of the puzzle---"NUMBER(g^)" rows by "NUMBER(g[0])"
   columns.  The elements of the array may be updated even though "g"
   is READONLY; e.g., "g[2,4]:=blk".  At the time "Solve" is called,
   the value of every array element is undefined.

   The first element of all "REF ARRAY" arrays ("g", "across", and
   "down") is at index 0.
*)

PROCEDURE Solve (READONLY g: Grid;  READONLY across, down: REF ARRAY OF TEXT)
    RAISES {no_solution};

END CrossWord.
```

You are to write the implementation of this module called `CrossWord.m3`. Notice that this is not a main program. To execute the `Solve` procedure you need a main program that calls `Solve`. You can use `Main.m3` that is provided for you, or write your own. Our main program `Main.m3` looks roughly like this:

```
MODULE Read EXPORTS Main;

IMPORT Wr, Stdio;
IMPORT CrossWord;

VAR
    g,n: CrossWord.Grid;
    a,d: REF ARRAY OF TEXT;
BEGIN
    g := GetGridFromStream (Stdio.stdin);  (* Read puzzle from std input *)
    GetAD (g,a,d);                         (* Form down/across words     *)
    n := NEW (CrossWord.Grid, NUMBER(g^), NUMBER(g[0]));
    TRY
        CrossWord.Solve (n, a, d);
        IF EqualGrid (g,n) THEN
            Wr.PutText (Stdio.stdout, "Solution found.\n");
        ELSE
            Wr.PutText (Stdio.stdout, "Unequal solution found!\n");
        END;
    EXCEPT
        CrossWord.no_solution => Wr.PutText(Stdio.stdout,"No solution found!\n");
    END;
END Read.
```

We will compile your program `CrossWord.m3` with our main program and the interface file `CrossWord.i3`. It is important that you insure the following command produces an executable file:

```
m3 -make Main.m3 CrossWord.i3 CrossWord.m3
```

We will test your program on several crossword puzzles. Your program should produce no output; it should just fill in the grid. Do not leave debug statements in your program.