

CS 2073, Computer Programming with Engineering Applications
Spring 1992
Final Examination

- (10) 1. Suppose an array of integers has been declared

```
A: array[1..100] of integer;
```

Suppose the locations from 1 to N have values stored in them. Write a code segment that will look up an integer X in this array. The segment should set a variable K equal to the location of X in the array, or to 0 in case X does not occur in the array.

- (10) 2. Consider a user-defined (or enumerated) type, with corresponding declarations:

```
type colors = (red, blue, green, yellow, purple, black, white);
var color: colors;
```

- (a) Is it legal to write

```
color := purple;
```

in a program?

- (b) In case `color` has the value `blue`, what will `ord(color)` be equal to?

- (c) In case `color` has the value `blue`, what will `succ(color)` be equal to?

- (10) 3. Write a segment of code that will evaluate the following expression:

$$(h)[\sin(a) + \sin(a+h) + \sin(a+2h) + \dots + \sin(a + (n-1)h) + \sin(a + nh)],$$

where $n = 100$, $a = 0$, $b = \pi$, and $h = (b-a)/n$. (Notice that $a + n*h$ just equals b .) In terms of calculus or areas or integrals, what does this sum represent?

- (10) 4. Give two of the more interesting facts about the C language that you learned from the lecture on C in class (or from class notes). (For full credit you need something not completely trivial about C.)

- (20) 5. Suppose you have declarations like those for the Laplace programming assignment (Number 6):

```
const (* below, m's used for rows, n's for columns *)
m1 = 20; (* top of inner space at 212 degrees*)
m2 = 40; (* bottom of inner space at 212 degrees*)
m3 = 60; (* lower boundary at 32 degrees *)
m0 = 30; (* start of liquid on the outside "bath" at 32 degrees*)
n1 = 20; (* left side of inner space at 212 degrees *)
n2 = 40; (* right side of inner space at 212 degrees *)
n3 = 60; (* far right boundary *)
```

```

type atype = array[0..m3+1, 0..n3+1] of real; (* rows and columns *)
      (* assumes that [0,0] is upper left corner *)
      ctype = array[0..m3+1, 0..n3+1] of char; (* matching character array *)
var a: atype;
    c: ctype;

```

(a) Write a procedure `initialize` with one parameter of type `ctype` that will initialize *all* the characters of its parameter to blanks. Does this parameter need to be a `var` (or reference) parameter?

(b) Write a procedure `stars` with two parameters, one of type `atype` and one of type `ctype`. `stars` should store a star `'*'` in each position `[i, j]` of the `c` array in case the temperature in the corresponding position in the `a` array is greater than 100.

- (20) 6. Recall the random number generator `random` that we used in class. In order to use it, you need the following declarations and initializations:

```

var seed: double;
function random(var seed:double): double;
(* the rest of random here *)
begin (* main program *)
    seed := 474747.0; (* or any value from 1.0 to 2^31-1 *)
    (* now make use of random *)
end.

```

(a) Write a segment of code which will use this random number generator to print 100 random real numbers between 0.0 and 1.0.

(b) Write a function `DICE` which will use this generator to simulate rolling two dice. The function should return the integer representing the total number of spots rolled. (A number between 2 and 12 inclusive.)

(c) Write a code segment which will keep track of the numbers of spots rolled, using an array declaration:

```
var SPOTS: array[2..12] of integer;
```

Each array element is a counter for that number of spots, so that if you get 5 from `DICE`, the array element `SPOTS[5]` should be incremented. The segment should print out the counts after 1000 simulated rolls. The output should look roughly like:

```

SUM OF 1: 28
SUM OF 2: 55
    ...
SUM OF 12: 30

```

- (20) 7. Consider the following Pascal program that handles character strings of varying lengths:

```
program strings;
const Maxstr = 10;
type
  basicstring = array[1..Maxstr] of char;
  stringtype =
    record
      str: basicstring;
      len: integer
    end;
var r, s, t: stringtype;
    n, k: integer;

procedure readstring(var s: stringtype);
var ch: char;
    i: integer;
begin
  i := 0;
  while not eoln and (i < Maxstr) do
  begin
    i := i + 1;
    read(ch);
    s.str[i] := ch
  end;
  s.len := i;
  readln
end;

procedure writestring(s: stringtype);
var i: integer;
begin
  write('');
  for i := 1 to s.len do
    write(s.str[i]);
  writeln('')
end;

function length(s: stringtype): integer;
begin
  (* the code for this function goes here *)
end;

procedure substring(s: stringtype;
  n: integer; k: integer; var t: stringtype);
var i: integer;
begin
  if (n + k - 1 > s.len) or (n <= 0) then
    writeln('*** error in substring ***')
  else
  begin
    for i := 1 to k do
      t.str[i] := s.str[i + n - 1];
    t.len := k
  end
end;

procedure substring(r, s: stringtype; var t: stringtype);
begin
  (* the code for this procedure goes here *)
end;

begin (* main program *)
```

```
writeln('Please enter a string on one line');
readstring(r);
writestring(r);                (* question (c) *)
writeln('Length:',length(r));  (* question (c) *)
writeln('Enter a starting position for a substring');
readln(n);
writeln('Enter a length for a substring');
readln(k);
substring(r, n, k, t);
writestring(t);                (* question (d) *)
writeln('Length:', length(t)); (* question (d) *)
writeln('Please enter a second string on one line');
readstring(s);
concat(r, s, t);
writestring(t)
end.
```

- (a) Which of the two basic methods for representing varying length strings are employed here? (Special character at the end, or store the length explicitly?)
- (b) Fill in the code for the function `length` above, so that it will correctly return the length of a string.
- (c) Suppose you type in "wagner" followed by a return in response to the "Please enter a string on one line" message. Exactly what will be stored in the global record structure named `r`? What will be printed out by the `writestring` and the following `writeln`? (These are marked as "question (c)" above.)
- (d) Suppose you enter 3 for a starting position (the value of `n`) and 2 for a length (the value of `k`). After the call to `substring`, what will be printed out by the `writestring` and the following `writeln`? (These are marked as "question (d)" above.)
- (e) The `concat` procedure should just tack `r` and `s` together to form a new long string named `t`. Give the code for this procedure.