

# CPSC 231 Practice final exam questions for winter 2006

## Part I: Short answer

### Question 1:

Trace the output of the following program.

```
program list (output);
```

```
const
```

```
    MAX = 3;
```

```
type
```

```
    NodePointer = ^Node;
```

```
    Node = record
```

```
        data : integer;
```

```
        next : NodePointer;
```

```
    end;
```

```
procedure add (var head : NodePointer;  
              newNode : NodePointer);
```

```
var
```

```
    temp : NodePointer;
```

```
begin
```

```
    if (head = NIL) then
```

```
        head := newNode
```

```
    else
```

```
        begin
```

```
            temp := head;
```

```
            while (temp^.next <> NIL) do
```

```
                temp := temp^.next;
```

```
            temp^.next := newNode;
```

```
        end;
```

```
        newNode^.next := NIL;
```

```
end;
```

```
procedure initialize (var head: NodePointer);
```

```
var
```

```
    i : integer;
```

```
    temp : NodePointer;
```

```
begin
```

```
    new(temp);
```

```
    for i := 1 to MAX do
```

```
        begin
```

```
            temp^.data := i;
```

```
            add(head,temp);
```

```
        end;
```

```
end;
```

```
procedure display (head : NodePointer);
```

```
begin
```

```
    while (head <> NIL) do
```

```
        begin
```

```
            write(head^.data);
```

```
            head := head^.next;
```

```
        end;
```

```
end;
```

```

begin
  var head : NodePointer;
  head := NIL;
  initialize(head);
  display(head);
end.

```

<< Write your answer here >>

3

## Question 2:

In the space provided below you are to show the output of the following program.

```

program list2 (output);
const
  MAX = 4;
type
  NodePointer = ^ Node;
  |
  Node = record
    data : integer;
    next : NodePointer;
  end;

procedure initialize (var head : NodePointer);
var
  i : integer;
  temp : NodePointer;
begin
  i := 1;
  if (head = NIL) then
    begin
      new(head);
      head^.data := i;
      head^.next := NIL;
    end;
  for i := 2 to MAX do
    begin
      new(temp);
      temp^.data := i;
      temp^.next := head;
      head := temp;
    end;
end;

procedure display (head : NodePointer);
begin
  while (head <> NIL) do
    begin
      write(head^.data, ' ');
      head := head^.next;
    end;
  writeln;
end;

```

```

begin
  var head : NodePointer;
  head := NIL;
  initialize(head);
  display(head);
end.

```

<< Write the output to the above program here >>

**4 3 2 1**

### Question 3:

For the following program you are to assume that the following declarations have already been made:

```

program recordExample (output);
const
  MAX      = 10;
  LENGTH   = 24;
type
  Student = record
    name : array [1..LENGTH] of char;
    id   : integer;
  end;
  StudentList = array [1..MAX] of Student;

  :      :      :
begin
  var aStudentList : StudentList;
  initialize(aStudentList);
  display(aStudentList);
end.

```

You can assume that the array has been properly initialized. Write a display procedure that will display all the elements of the array from the first element to the last with the information for each student on a separate line and each field should be separated by a space.

```

procedure display (aStudentList : StudentList);
var
  i : integer;
begin
  for i := 1 to MAX do
  begin
  writeln(aStudentList[i].name, ' ', aStudentList[i].id);
  end;
end;

```

**Question 4:**

You are to write a program that will draw a filled square within a 10x10 character array. The initial value for each cell of the array is a dot “.”. A square will be drawn within the array by changing some of the elements to the number sign (see the example of the program running below). The program will get the information for the position and size of the square by prompting the user for the starting row and column (r, c) coordinates and for the length of a side.

For example if the user types in 2, 2 for the starting row and column coordinate values and 4 for the length, the following square would be drawn:

**Example execution**

```
Enter row coordinate of top left corner: 2
Enter column coordinate of top left corner: 2
Enter the length of a side: 4
```

```
. . . . .
. # # # # . . . . .
. # # # # . . . . .
. # # # # . . . . .
. # # # # . . . . .
. . . . .
. . . . .
. . . . .
. . . . .
. . . . .
```

Most of the program has already been completed; you need only to implement the “fillInSquare” procedure, which fills in the appropriate parts of the array. You can assume that the values entered by the user will not result in a square that exceeds the bounds of the array.

```
program drawSquare (input, output);
type
  CharacterArray = array[1..10,1..10] of char;

procedure initializeSquare (var arr : CharacterArray);
(* Initializes the whole array to dots. *)
  :      :      :      :

procedure getSquareData (var topRow   : integer;
                        var topColumn : integer;
                        var length    : integer );
begin
  write('Enter row coordinate of top left corner: ');
  readln(topRow);
  write('Enter column coordinate of top left corner: ');
  readln(topColumn);
  write('Enter the length of a side: ');
  readln(length);
end;
```

```

procedure fillInSquare (  topRow    : integer;
                        topColumn  : integer;
                        length     : integer;
                        var arr     : CharacterArray);

```

```

var
  r : integer;
  c : integer;
begin
  for r := topRow to (topRow + length - 1) do
  begin
    for c := topColumn to (topColumn + length - 1) do
    begin
      arr[r][c] := '#';
    end;
  end;
end;

```

```

procedure displaySquare (arr : CharacterArray);

```

```

var
  r : integer;
  c : integer;
begin
  for r := 1 to 10 do
  begin
    for c := 1 to 10 do
    begin
      write(arr[r][c]);
    end;
    writeln;
  end;
end;

```

```

begin
  var topRow    : integer;
  var topColumn : integer;
  var length    : integer;
  var arr       : CharacterArray;

```

```

  initializeSquare(arr);
  getSquareData(topRow, topColumn, length);
  fillInSquare(topRow, topColumn, length, arr);
  displaySquare(arr);
end.

```

**Short answer question #5**

Trace the output of the following program

```

program arrayExample (input, output);
begin
  var grid : array [1..4, 1..4] of integer;
  var r    : integer;
  var c    : integer;

  for r := 1 to 4 do
  begin
    for c := 1 to 4 do
    begin
      grid[r][c] := r;
    end;
  end;

  for r := 1 to 4 do
  begin
    for c := 1 to 4 do
    begin
      write(grid[c][r]);
    end;
    writeln;
  end;
end.

```

<< Put your answer here >>

1234  
 1234  
 1234  
 1234

**Part I: Multiple choice**

Select the answer that *best* answers the question. Unless otherwise stated you can assume that all programs and program fragments will compile.

1. Which of the following statements are true of the array declaration shown below?

```
var arr : array[1..3,1..4] of char;
```

- a) arr has 3 rows (index 1-3) and 4 columns(index 1-4)
- b) arr has 3 rows (index 0-2) and 4 columns (index 0-3)
- c) arr has 4 rows (index 0-3) and 3 columns (index 0-2)
- d) arr has 4 rows (index 1-4) and 3 columns (index 1-3)
- e) None of the above statements are true

**Answer: a**

For multiple choice questions 2 – 4 please refer to the following program.

```
program fileIO (output);
begin
  var ch   : char;
  var data : text;

  reset(data, 'data');
  while NOT EOF (data) do
  begin
    read(data, ch);
    write(ch);
    read(data, ch);
    write(ch);
    readln(data, ch);
    write(ch);
  end;
  close(data);
end.
```

And the following versions of the data file called “data.txt”:

“data.txt” – version 1  
The data file is empty

“data.txt” – version 2  
123<EOL>123<EOL>

“data.txt” – version 3  
123<EOL>123456<EOL>1

Note: EOL stands for the “end of line” marker.

2. What will be output of running the program with “data” version 1?
  - a. Nothing will be displayed
  - b. 123123
  - c. 1231234561
  - d. 123<EOL>123456<EOL>1
  - e. 1231231./a.out: attempt to read past end of file `data' (error #454 at 11d5b)

**Answer: a**

3. What will be output of running the program with “data” version 2?
  - a. Nothing will be displayed
  - b. 123123
  - c. 1231234561
  - d. 123<EOL>123456<EOL>1
  - e. 1231231./a.out: attempt to read past end of file `data' (error #454 at 11d5b)

**Answer: b**

4. What will be output of running the program with “data” version 3?

- a. Nothing will be displayed
- b. 123123
- c. 1231234561
- d. 123<EOL>123456<EOL>1
- e. 1231231./a.out: attempt to read past end of file `data' (error #454 at 11d5b)

**Answer: e**

5. Which of the following are true of the assignment statement “intPtr^ := 10;”?
- a) It puts the value 10 in an integer variable called ‘intPtr’.
  - b) It makes the pointer point to memory address ten.
  - c) It dereferences a pointer and should store in the dynamically allocated memory the integer ten.
  - d) (a) & (b)
  - e) None of the above.

**Answer: c**

For questions 6 – 9 please refer to the program:

program pointerQuestion (output);

type

IntegerPointer = ^integer;

procedure proc (ptr1 : IntegerPointer;  
ptr2 : IntegerPointer);

var

temp : IntegerPointer;

begin

new(temp);

temp^ := 888;

ptr1 := temp;

ptr2 ^:= temp^;

writeln('#3: ', ptr1^, ' ', ptr2^);

end;

begin

var ptr1 : IntegerPointer;

var ptr2 : IntegerPointer;

new(ptr1);

new(ptr2);

ptr1^ := 1;

ptr2^ := 2;

writeln('#1: ', ptr1^, ' ', ptr2^);

ptr1^ := ptr2^;

writeln('#2: ', ptr1^, ' ', ptr2^);

proc(ptr1, ptr2);

writeln('#4: ', ptr1^, ' ', ptr2^);

end.

6. What will be the output of the statement: writeln('#1: ', ptr1^, ' ', ptr2^);



- a. #1: 1 1
- b. #1: 1 2
- c. #1: 2 1
- d. #1: 2 2
- e. None of the above

**Answer: b**

7. What will be the output of the statement: `writeln('#2: ', ptr1^, ' ', ptr2^);`
- a. #2: 1 1
  - b. #2: 1 2
  - c. #2: 2 1
  - d. #2: 2 2
  - e. None of the above

**Answer: d**

8. What will be the output of the statement: `writeln('#3: ', ptr1^, ' ', ptr2^);`
- a. #3: 2 2
  - b. #3: 2 888
  - c. #3: 888 2
  - d. #3: 888 888
  - e. None of the above

**Answer: d**

9. What will be the output of the statement: `writeln('#4: ', ptr1^, ' ', ptr2^);`
- a. #4: 1 2
  - b. #4: 2 2
  - c. #4: 2 888
  - d. #4: 888 888
  - e. None of the above

**Answer: c**

JT: Good luck with the real thing!

