

UNIVERSITY OF SWAZILAND

FACULTY OF SCIENCE

DEPARTMENT OF ELECTRICAL & ELECTRONIC ENGINEERING

PROGRAMMING TECHNIQUES I

COURSE CODE – EE271

SUPPLIMENTARY EXAMINATION

JULY 2011

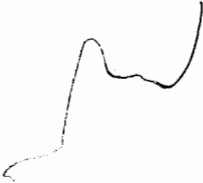
DURATION OF THE EXAMINATION - 3 HOURS

---

**INSTRUCTIONS TO CANDIDATES**

1. There are **FIVE** questions in this paper. Answer any **FOUR** questions only.
2. Each question carries equal marks.
3. Use correct notation and show all your steps clearly in any program analysis.
4. All programs should be well documented and indented.
5. Start each new question on a fresh page.

**DO NOT OPEN THIS PAPER UNTIL PERMISSION HAS BEEN GIVEN BY THE INVIGILATOR.**



### Question 1

- a) Explain the rationale for using iterative control structures in C programming. Give three examples of iterative control structures. For each control structure given as an example, explain its semantics. [12]
- b) Explain the difference between *semantic* and *syntactic* programming errors. Give one example of each. [3]
- c) Explain the following terms as used in programming languages: [3]
  - (i) Machine Language
  - (ii) Assembly Language
  - (iii) High Level Language
- d) Explain the following terms in the C programming language:
  - (i) Variable [4]
  - (i) Variable Scope [3]

### Question 2

Assume that reading is from the keyboard and output is displayed on the screen. Also assume that the following declarations have already been given.

```
struct product {  
    char name[30];  
    int price;  
} inventory[5];
```

Write executable C statements with proper syntax (not a complete program), to perform each of the following tasks independently. Use only the declarations given above.

- (i) Enable the user to capture a list of five products and their values into an inventory list. [2]
- (ii) Display the matching range of each product according to the following criteria: If the price of the product is E10-E50 display "LOW COST"; if the value is greater than E50 but less than or equal to E150, display "MEDIUM COST"; if the value is greater than E150 but less than or equal to E400, display "HIGH COST"; otherwise display "UNKNOWN PRODUCT COST". [5]
- (iii) Display the most expensive product in the list. [5]
- (iv) A function to sort the product inventory list in ascending order according to product prices. [8]
- (v) Assuming an inventory list of 50 products, display all the products in the 'LOW COST' and 'HIGH COST' categories according to the criteria in (ii). [5]

### Question 3

Answer each of the following. Assume that single-precision floating point numbers are stored in 4 bytes, and that the starting address of the array is at location 1002500 in memory. Each part of this question should use the results of previous parts where appropriate.

- a. Define an array of type float called *numbers* with 10 elements, and initialize the elements to the values 0.0, 1.1, 2.2, 3.3, 4.4, 5.5, 6.6, 7.7, 8.8, 9.9. Assume the symbolic constant *SIZE* has been defined as 10. [2]
- b. Define a pointer, *nPtr*, that points to an object of type float. [1]
- c. Print the elements of array *numbers*. Print each number with 1 position of precision to the right of the decimal point. [4]
- d. Give two separate statements that assign the starting address of array *numbers* to the pointer variable *nPtr*. [3]
- e. Print the elements of array *numbers* using pointer/offset notation with the pointer *nPtr*. [5]
- f. Assuming that *nPtr* points to the beginning of array *numbers*, what address is referenced by *nPtr + 8*? What value is stored at that location? [3]
- g. Assuming that *nPtr* points to *numbers[5]*, what address is referenced by *nPtr-4*. What is the value stored at that location. [7]

#### Question 4

Write a C program to compute and print a Pascal triangle. Your program should take as input the number of lines to be computed and printed *n*. Your program should give the user four options for printing the triangle: *left*, *right*, *up*, and *down*. For example with the *down* option selected for printing and *n = 5* lines, the output should be as shown below. Your program should be able to compute and print up-to 15 rows of the triangle.

```

1 4 6 4 1
  1 3 3 1
    1 2 1
      1 1
        1

```

Your program will be graded according to the following criteria:

- i. *Correctness* – does the program produce the desired result i.e. for a specified number of lines and printing direction, can the program produce the correct Pascal triangle. [15]
- ii. *Clarity* - proper indentation of program makes it easy to read. [2]
- iii. *Sensible naming of variables* – make it easy to understand code when debugging. [3]
- iv. *Proper use of comments* – comments also make the program easy to understand. [2]
- v. *Program modularity*- has the programmer made good use of functions for performing specific tasks in the program? [3]

#### Question 5

Carefully analyse the program shown in Figure 5 and determine its output. Show all working. [25]

```

#include <stdio.h>
#include <conio.h>
#include <math.h>

/* Function prototypes */
void function1();
void function2();
void function3();
void function4();

/* Global variables */
int M[5][5] = {0};
int r;
int c;

int main (void){
    function1();
    function2();
    function3();
    function4();
    printf("Press any key to continue.....\n");
    while(!kbhit()) { }
    return 0;
}

void function1() {
    for(r=0; r<5; r++) {
        for(c=0; c<5; c++) {
            M[r][c] = 2*r*r + 3*r*c + 4*c*c;
        }
    }
}

void function2() {
    r=0;
    while(r<5) {
        for(c=0; c<5; c++) {
            if(M[r][c]%2==0) {
                M[r][c] = sqrt(M[r][c]/2);
            } else {
                M[r][c] = pow(3,M[r][c]%2);
            }
        }
        r++;
    }
}

void function3() {
    for(r=0; r<5; r++) {
        for(c=0; c<5; c++) {
            if(r==c) {
                M[r][c] = M[r][c]*r*c;
            }
        }
    }
}

void function4() {
    for(r=0; r<5; r++) {
        for(c=0; c<5; c++) {
            if(r>=c) {
                printf("%d ", M[r][c]);
            } else {
                printf(" ");
            }
        }
        printf("\n");
    }
    for(r=3; r>=0; r--) {
        for(c=0; c<5; c++) {
            if(r>=c) {
                printf("%d ", M[r][c]);
            } else {
                printf(" ");
            }
        }
        printf("\n");
    }
}
}

```

Figure 5. Program for Question 5

END OF EXAM PAPER