

**Question 1 (COMPULSORY) [60 marks]**

- (a) **False**
- (b) **True**
- (c) **True**
- (d) **False**
- (e) **first**
- (f) **value is 2.235**
- (g) **i is 13  
i is 2  
i is 0**
- (h) **condition is true**
- (i) **True**
- (j) **value is 2.20**
- (k) **j is -4  
j is -3  
j is -2  
j is -1  
j is 0**
- (l) **result is -675**
- (m) **(m-3) \*arr[0]\*\*arr[2]**
- (n) **True**
- (o) **(o-3) fn() takes 1 argument of type pointer-to-float and does not return a value.**
- (p) **(p-2) if it exists, open datafile.txt for reading only, otherwise return an error.**
- (q) **string=YbcdYfghYjklmn**
- (r) **False**
- (s) **x[0] is 2.00  
x[1] is 1.00  
x[2] is 0.67  
y[0] is 2  
y[1] is 1  
y[2] is 0**
- (t) **(t-3) emp1.age = 30;**

**Question 2 [40 marks]**

(a) The following fragment of C code outputs the digits in a string called string to the screen using a for loop:

```
int i;
char string[500];
/* assume string[] is somehow filled with letters, digits, etc */
for (i=0; string[i]!='\0'; i++){
    if ((string[i] >= '0') && (string[i] <= '9')){
        printf("%c", string[i]);
    }
}
```

Re-write this code fragment using a while loop instead of the for loop.

```
int i; /* could be: int i=0; */
char string[100];
/* assume string[] is somehow filled with letters, digits, etc */
i=0;
while (string[i]!='\0'){
    if ((string[i] >= '0') && (string[i] <= '9')){
        printf("%c", string[i]);
    }
    i++;
}
```

(b) The following fragment of C code uses if statements to determine whether to Sell, Hold, or Buy a stock based on its price:

```
if (stock_price >= 50){
    printf("Sell\n");
}
if ((stock_price >= 25) && (stock_price < 50)){
    printf("Hold\n");
}
if (stock_price < 25){
    printf("Buy\n");
}
```

Re-write this code fragment using if/else-if/else statements instead of the if statements.

```
if (price >= 50){
    printf("Sell\n");
} else if (price >= 25){
    printf("Hold\n");
} else {
    printf("Buy\n");
}
```

(c) Consider the following C program:

```
#include <stdio.h>
/* DEFINITION OF FUNCTION "num_pos" GOES HERE */
void main(void) {
    int i, j=0, array1[8]={1,-1,-1,0,0,1,0,-1};
    j = num_pos(array1,8); /* function call to num_pos() */
    printf("there are %d positive elements of array1[]", j);
}
```

Write down the definition of function num\_pos() which counts the number of positive elements in its input array, so that the output of the above program is:

there are 2 positive elements of array1[]

```
int num_pos(int array[], int size){
    int i, k=0; /* i is a local loop counter */
    for (i=0; i<size; i++){
        if (array[i]>0){
            k++;
        }
    }
    return (k);
}
```

**Question 3 [40 marks]**  
**Answer parts (a) and (b).**

(a) Consider the following C program:

```
#include <stdio.h>
void main(void)
{
    int intarr[3], i;
    for (i=0 ; i<3; i++){
        printf("enter value number %d: ", i+1);
        scanf("%d", &intarr[i]);          /* LINE 1 */
        printf("you entered %d\n", intarr[i]); /* LINE 2 */
    }
}
```

Re-write the lines LINE 1 and LINE 2 using “array pointers” instead of array subscripts.

LINE 1: `scanf("%d", intarr+i);`  
LINE 2: `printf("you entered %d\n", *(intarr+i));`

(b) Consider the following C program:

```
#include <stdio.h>
void main(void)
{
    FILE *fptr1, *fptr2;
    int inp;
    fptr1 = fopen("input.dat", "r");
    fptr2 = fopen("output.dat", "w");
    while (fscanf(fptr1, "%d", &inp)==1){
        if (inp >= 0){
            fprintf(fptr2, "%d ", inp);
        }
    }
    fclose(fptr1);
    fclose(fptr2);
}
```

(i) Suppose the file input.dat contains the following data:

0  
-2  
2  
6  
-1  
-5

Complete this sentence:

After executing this program, the file output.dat contains \_\_\_\_\_ .

After executing this program, the file output.dat contains 0 2 6

(ii) Re-write the above program so that it writes all values between -1 and 4 (inclusive) in the file input.dat to a new file called output2.dat

```
#include <stdio.h>
void main(void)
{
    FILE *fptr1, *fptr2;
    int inp;
    fptr1 = fopen("input.dat", "r");
    fptr2 = fopen("output2.dat", "w");
    while (fscanf(fptr1, "%d", &inp)==1){
        if (inp >= -1 && inp <= 4){
            fprintf(fptr2, "%d ", inp);
        }
    }
    fclose(fptr1);
    fclose(fptr2);
}
```