ESC 101: Fundamentals of Computing **Major Quiz 2** Date: 01 – 04 - 2019

Name	**ANSWER KEY**					**A**
Roll No.		Dept.		Section		

Total 55 Marks

Instructions:

1. This question paper contains a total of **4** pages (**7** printed side of paper).
2. Write your roll number on every sheet of this booklet.
3. Write final answers neatly with a blue/black pen in the given boxes.
4. **Answers written outside the box will NOT be graded.**

Q. 1. Write TRUE or FALSE against the given statements (1 x 10 = 10 Marks)

1	If we allocate a memory of 100 bytes using a single malloc call, we can free 50 bytes from this block using free().	**FALSE**
2	Size of a union variable is equal to sum of all the sizes of constituent variable in the union.	**FALSE**
3	When a structure is passed as an argument to a function, it is passed by reference .	**FALSE**
4	When an array is passed as an argument to a function, it is passed by reference.	**TRUE**
5	Let int a[10] be an array, then the pointer b, initialized as,int *b = a++ will point to a[1].	**FALSE**
6	Malloc returns a pointer to a memory location which is of the type void*.	**TRUE**
7	Size of int pointer is 4 bytes, char pointer is 1 byte and long pointer is 8 bytes.	**FALSE**
8	An 2D array (int a[10][10]) must have a contiguous block of (10*10*sizeof(int)) bytes in the memory.	**TRUE**
9	We can use free to release memory of a statically declared array. (For eg. int a[10]; free(a);).	**FALSE**
10	A segmentation fault refers to a situation when your program tries to access/modify memory location not allocated to it.	**TRUE**

Q. 2. Multiple Choice Question (Single Correct Only) (2 x 5 = 10 Marks)

1. During memory allocation, automatic memory allocation and static memory allocation respectively use

A) stack and heap [ ]
B) heap and stack [ ]
C) heap and heap [ ]
D) stack and stack [●]

2. The function free() can be used on which of the following types of memory

A) statically allocated memory [ ]
B) dynamically allocated memory [●]
C) automatically allocated memory [ ]
D) both (b) & (c) [ ]

3. Which of the following can cause segmentation faults

A) dereferencing a dangling pointer [●]
B) printing the value of a dangling pointer [ ]
C) modifying the value of a dangling pointer [ ]
D) both (a) & (c) [ ]

4. Linked lists are FASTER than arrays for which of the following operations. Assume that only the head of the linked list and the name of the array are given.

A) inserting an element (pointer to the node after which insertion is to be done is given) [●]
B) searching for an element [ ]
C) modifying an element given its position from the start/head [ ]
D) all of the above [ ]

5. Which of the following statements is correct?

A) array name variables cannot be reassigned [●]
B) having a return statement in a void function can cause a run time error [ ]
C) calloc() returns a integer pointer to the memory area alloted which has all its values assigned zero [ ]
D) none of the above [ ]

Q. 3. Multiple Choice Question (Multiple Correct) **(3 x 5 = 15 Marks)**

1. Consider the following 2 Dimensional array int a[20][20]; Choose all the option(s) which is equivalent to a[5][10].

A) (*(a + 5) + 10) [ ]
B) *(*(a + 5) + 10) [●]
C) (*(a + 5))[10] [●]
D) **(a + 5*20 + 10) [ ]
E) *(*a + 5*20 + 10) [●]

2. Consider the following structure and the following two instantiations.

```
struct student {
      char *name;
      int class;
      int roll_number;
}
```

Instantiations:

```
struct student *s = (struct sudent*) malloc(sizeof(student));
struct student t;
```

Choose the correct option(s) w.r.t. above instantiations of struct student.

Roll No. ______________

A) s will point to a memory block of size = 2*sizeof(int) + sizeof(char*). [●]

B) s.roll_number will access the roll_number field of the student structure instantiation s. [ ]

C) t.roll_number will access the roll_number field of the student structure instantiation t. [●]

D) s->name will access the name field of the student structure instantiation s. [●]

E) t->name will access the name field of the student structure instantiation t. [ ]

3. Consider the following string: char a[100] = "THE ANSWER IS\0 42"; Choose all the correct option(s):

A) strlen(a) is equal to 13. [●]

B) printf("%c", *a); will print: T [●]

C) After strcat(a, a), printing a will print THE ANSWER ISTHE ANSWER IS [●]

D) Memory used by array a is equal to 13*sizeof(char) [ ]

E) Printing a[50] may give segmentation fault. [ ]

4. Consider a singly linked list with 10 elements. Where each node is designed as follows.

```
struct node {
        struct node *next;
        int value;
}
```

Choose the correct option(s) with respect to above linked list. Assume that we DON'T have the head pointer, unless stated otherwise.

A. If we have a pointer to 4th element, we can delete 5th element. [●]

B. If we have a pointer to 5th element, we can delete 4th element. [ ]

C. If we have head pointer, head->value will be equal to value of the first element. [●]

D. If we have head pointer, size of the head pointer will be equal to sizeof(char*). [●]

E. If we have head pointer, head + 1 will point to the second element of the list. [ ]

5. Consider the following code snippet. Choose all the correct option(s).

```
int main() {
      int a = 10, b = 20;
      int *pa = &a, *pb = &b;
      int **ppa = &pa, **ppb = &pb;

      printf("%d\n", (*pa)*(*pb));          // (1)
      ppa = ppb;
      printf("%d\n", (*pa)*(**ppa));        // (2)
      printf("%d\n", (*ppa == *ppb));       // (3)
      *pa = b;
      printf("%d\n", (**ppa != *pa));       // (4)
}
```

A) 200 will be printed at line (1) [●]

B) 200 will be printed at line (2) [●]

C) 1 will be printed at line (3) [●]

D) 1 will be printed at line (4) [ ]

E) None of the above [ ]

Q. 4: Write the output for the following programs. **(5 x 2 = 10 Marks)**

1. MAGIC SQUARE

```c
#include <stdio.h>

void function(int size, int a[][3]){
   int sqr = size * size;
   int i = 0, j = 1, k;
   for (k = 1; k <= sqr; ++k) {
       a[i][j] = k;
       i--;
       j++;

       if (k % size){
           if (j == size) j -= size;
           else if (i < 0) i += size;
       }
       else {
           i += 2;
           j--;
       }
   }
   for (i = 0; i < size; i++){
       for (j = 0; j < size; j++){
           printf("%d  ", a[i][j]);
       }
       printf("\n");
   }
   printf("\n");
}

int main( ){
   int arr[3][3];
   function(3, arr);
   return 0;
}
```

OUTPUT

```
8 1 6
3 5 7
4 9 2
```

2. RECURSIVE DISPLAY

Roll No. ____________________

```
#include <stdio.h>
#include <stdlib.h>

typedef struct node{
   int data;
   struct node *next;
}node;

void print1(node * );
void print2(node * );

void insert_new_node (node ** head_ref, int new_data){
   node * new_node = (node *) malloc(sizeof(node));
   new_node -> data = new_data;
   new_node -> next = (*head_ref);
   (*head_ref) = new_node;
}

void print1(node *head){
   if (head == NULL) return;
   print2(head -> next);
   printf("%d ", head -> data);
}

void print2(node *head){
   if (head == NULL) return;
   printf("%d ", head -> data);
   print1(head -> next);
}

int main (){
   node *head = NULL;
   insert_new_node (&head, 1);
   insert_new_node (&head, 2);
   insert_new_node (&head, 3);
   insert_new_node (&head, 4);
   printf ("List : ");
   print1(head);
   return 0;
}
```

OUTPUT

```
List : 3 1 2 4
```

Q. 5: Fill in the blanks with appropriate code in the following programs. (5 x 2 = 10 Marks)
Assume appropriate includes.

1. Given an area of n x m. You have infinite number of tiles of size 2^i X 2^i where i = 0, 1, 2,... so on. The task is to find minimum number of tiles required to fill the given area with tiles.

```
#include<stdio.h>

int minTilesRequired(int n , int m){
    if (n == 0 || m == 0) return 0;
    else if (n%2 == 0 && m%2 == 0)
       return minTilesRequired(____A____, ____B___);
    else if (n%2 == _____C____ && m%2 == 1)
        return (n + minTilesRequired(n/2, m/2));

    else if (n%2 == _____D____ && m%2 == 0)
        return (m + minTilesRequired(n/2, m/2));

    else
        return (n + m - 1 + minTilesRequired(n/2, m/2));
}

int main()
{
    int n,m;
    scanf("%d%d",&n,&m);
    printf("%d", minTilesRequired(n,m));
    return 0;
}
```

A: **n / 2**

B: **m / 2**

C: **0**

D: **1**

Roll No. ____________________

2. In computer science, a STACK is an abstract data type that serves as a collection of elements with two principal operations, PUSH which adds an element to the collection and POP which removes the most recently added element that was not yet removed. This makes the stack a Last-In-First-Out (LIFO) data structure. In a LIFO data structure, the last element added to the stack will be the first one to be removed.

```
typedef struct node{
   int data;
   struct node *next;
}node;
node *head = NULL;

void push(int value){
   node *tmp = (node*)malloc(sizeof(node));     //(A)
   tmp -> data = value;
   tmp -> next = head;                          //(B)
   head = tmp;
}
int pop(){
   node *tmp = head;
   int n = tmp->data;                           //(C)
   head = head->next;
   free(tmp);
   return n;
}
void display(node *head){
   if(head == NULL) printf("NULL\n");
   else{
       printf("%d ", head -> data);
       display(head->next);                    // (D)
   }
}
int main(){
   push(10);
   push(20);
   push(30);
   display(head);
   pop();
   display(head);
   return 0;
}
```