United International University (UIU)

Dept. of Computer Science & Engineering (CSE)

Midterm Exam **Trimester: Spring - 2020**

Course Code: **CSI 217** Course Title: **Data Structures**

Total Marks: **30** Duration: **1 hour 45 minutes**

Answer all 6 questions. *Each question contains 5 marks.*

1. Answer the following questions on time complexity.
   a. Find out the **run time complexity** of the following algorithm. Clearly show the complexity of the necessary individual statements. [3]

```
1.  for ( i=0; i<n; i++ )
2.  {
3.      printf("Hello\n")
4.  }
5.
6.  for ( i=0; i<n; i++ )
7.  {
8.      for ( j=n; j>0; j=j/2 )
9.      {
10.         printf("World\n")
11.     }
12. }
```

   b. We already know that dynamic arrays expand on their own as more data is inserted. Clearly explain how the size of the dynamic array is increased in the memory. [2]

2. Answer the following questions on sorting algorithms.
   a. Perform **quick sort** on the following array till the end of the **second** partition. [3]

5	9	7	8	10	2	1	3	4

   b. Assume an array consisting of millions of integers and you know for sure that the data is not sorted. The array also contains a huge range of positive integers. How would each of the following sorting algorithms perform on the given array?

   Among, merge sort, quick sort and counting sort, which algorithm do you think would be the best choice? You have to clearly explain your answer. [2]

3.
   a. Suppose there is a **Singly Linked List,**
   Write a function to delete the middle element of the linked list. You can assume that the linked list always contains more than 3 nodes. [3]

   b. Suppose there is a **Circular Singly Linked List**,
   Given the pointer to the head of the circular singly linked list, write a function to add a node at the end of the linked list. The new node contains the value 10. [2]

4. Suppose there is a **doubly linked list node** declared as below:

```
struct node{
        int data;
        node * next;
        node * prev;
}
```

   a. Suppose you are given the pointer to the head of the doubly linked list, write a function to print the contents of the linked list in reverse order. [2]

   b. Show the effect of each of the statements by drawing nodes and linking them. [3]
      i. a = (node*) malloc(sizeof(node));
      ii. b = (node*) malloc(sizeof(node));
      iii. c = (node*) malloc(sizeof(node));
      iv. a->prev = c;
      v. a->next = NULL;
      vi. b->prev = a;
      vii. b->next = c;
      viii. c->prev = NULL;
      ix. c->next = a;
      x. free(b);

5.

   a. How would you delete the middle element in a STACK and keep the rest of the elements intact in their places? Assume that you already have the PUSH, POP and PEEK functions implemented. [3]

   b. Write the PUSH and POP functions of STACK using Linked List. [2]

6.

   a. We have already written a function to let the user see the element at the front of the queue. Now write a function to return the element at the back of a QUEUE. [1]

   b. Suppose a Stack is implemented using an array and a Queue is implemented using another array. Now consider the following sequences containing push, pop, peek function of stack and enqueue, dequeue and top function of queue. Assume that the size of stack is 4 and the size of the queue is also 4p.

      You have to draw what happens after performing each of the following operations. **Clearly show the value top of the stack and front and rear of queue in each step.** Also show the elements in the Stack and Queue after the operations: [4]

      push(10), enqueue(5), push(top()), push(top()), enqueue(peek()), enqueue(peek()), dequeue(), pop()

GOOD LUCK!