United International University (UIU)

Dept. of Computer Science & Engineering (CSE)

***Midterm Exam*:: Trimester: Summer - 2019**

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

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

Answer all 5 questions. *Numbers to the right of the questions denote their marks.*

1.

a. Find out the run time complexity of the following algorithm: [3]

```
int sum = 0;
int i=0;
while(i<n){
   sum=sum+i;
   printf("%d", i);
   i++;
  }
printf("%d\n", sum);
```

b. Fill up the following gap using the required number of element comparisons for the linear search and binary search. [2]

Array A: 10 20 30 40 50 60 70 80 90 100

	Key=10	Key= 70	Key=100	Key=35
Linear Search :	?	?	?	?
Binary Search:	?	?	?	?

2. Answer the following questions on Stack. You are **NOT** allowed to use any other data structure other than stack for the following questions. Assume all necessary functions of the stack are already implemented.

a. Write a function to reverse a stack. [1]

b. Write a function to input '**n**' integers from the user and push them into the stack in a sorted order, i.e. the largest element will be at the top and the second largest element will be below it and so on. The function will not return anything. [4]

3.

a. Consider the following array **A: {-20, 15, 37, 45, 88, 76, 89, 59, 97}**
Can we apply binary search on **A**? State your reasons [1]

b. If your answer in the previous question is no, then prepare **A** for binary search. Otherwise leave it as it is. Now manually trace binary search on **A** when you are searching for the value 77. Show the values of low (L), high (R), mid at each step. [2]

c. Consider the following structure:

```
struct student{
        char name[100];
        int id;
        double cgpa;
};
```

You are given an array **S** which contains 50 students, **sorted by id**. Given the **id** of a student, write a code to print the cgpa of that student. Make your code as efficient as possible. [2]

4.

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

```
struct student{
        char name[100];
        int id;
        double cgpa;
}
```

Given a pointer to the last node of a doubly linked list only, write a code to delete the middle node and the node previous to the middle node of the linked list. For this problem you can assume that the number of nodes in the linked list is odd and more than 20. [2.5]

b. Suppose there is a circular linked node declared as below:

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

Given a pointer to the **first** node of a circular linked list, write a code to **delete the first node** of the linked list. For this problem you can assume that the number of nodes in the linked list is more than 20. [2.5]

OR

Apply merge sort on the following data (You need to show all the steps): [5.0]

94	35	68	84	55	23	79	55	19	22

5.

a. Show the manual tracing of bubble **or** selection sort in the following array for sorting in the ascending order

A: {6, 5, 3, 1, 8, 7, 2, 4, 0, 10, 12} [3]

b. What is the name of the following sorting algorithm? Why do you think the following algorithm gives linear time complexity in an already sorted array? Please explain briefly. [2]

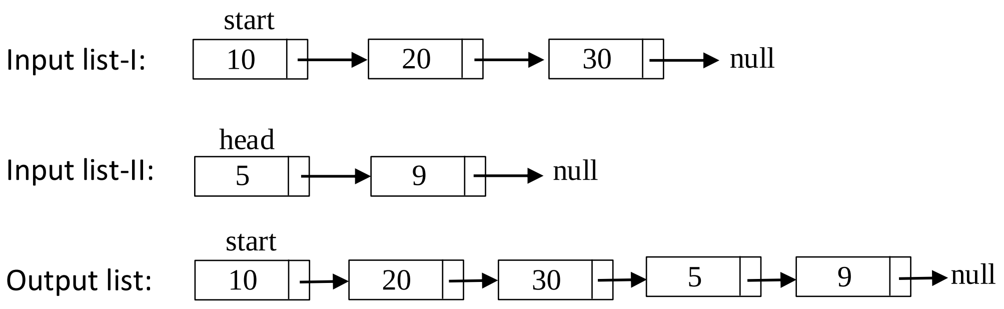
```
for $j = 2$ to $A.length$
    $key = A[j]$
    $i = j - 1$
    while $i > 0$ and $A[i] > key$
        $A[i + 1] = A[i]$
        $i = i - 1$
    $A[i + 1] = key$
```

6.

a. Suppose there is a singly linked node declared as below, where head is the pointer to the first node of the linked list

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

Write a code that will concatenate the two linear linked lists and display the list after concatenation. [2.5]



Output: 10, 20, 30, 5, 9

b. Using the structure of the Ques. 6a, show the effect of each of the statements by drawing nodes and linking them: [2.5]

a = (node*)malloc(sizeof(node));	**OR**	a = new node();
b = (node*)malloc(sizeof(node));	**OR**	b = new node();
c = (node*)malloc(sizeof(node));	**OR**	c = new node();
d = (node*)malloc(sizeof(node));	**OR**	d = new node();

a->value=2;

b->value=3;

c->value=4;

d->value=5;

d->next = a;

a->next = b;

b->next = c;

c->next = d;

e=a;

d->next = a->next;

free(a);