National University of Computer & Emerging Sciences, Karachi
Fall-2020 CS-Department
Midterm I (Sol)
October 19, 2020 Slot: 9 AM – 10 AM

Course Code: CS 218	Course Name: Data Structures
Instructor Name:	Muhammad Rafi / Dr. Ghufran/Basit Jasani/ Zain ul Hassan
Student Roll No:	Section No:

- Return the question paper.
- Read each question completely before answering it. There are **4 questions and 1 page, 2 sides.**
- In case of any ambiguity, you may make assumption. But your assumption should not contradict with any statement in the question paper.
- All the answers must be solved according to the sequence given in the question paper.
- Be specific, to the point while coding, logic should be properly commented, and illustrate with diagram where necessary.

Time: 60 minutes. **Max Marks**: 40 points

Object Oriented Programming	
Question No. 1	**[Time: 5 Min] [Marks: 5]**

What are the problems , if any, with the following constructor and destructor of class A?

```
int **ptr;  // A's Data member

A() {

  ptr = new int* [5];

  for (int i = 0; i < 5; i++)

    *(ptr+i) = new int[4];

}
```

```
~A() {

  delete[] ptr;

  for (int i = 0; i < 5; i++)

    delete ptr[i];

}
```

- Remember "Rule of Three"
- The memory release order should be exactly opposite of the memory acquired order.

```
int **ptr;  // A's Data member

A() {
  ptr = new int* [5];
  for (int i = 0; i < 5; i++)
    *(ptr+i) = new int[4];
}
```

```
~A() {

    for (int i = 4; i <=0; i--)
    delete[] ptr[i];
  delete[] ptr;

}
```

Recursion	
Question No. 2	**[Time: 10 Min] [Marks: 10]**

Write a recursive function **CountOnes** that takes an unsigned integer and return the number of ones in binary representation of that given number. For an example: if 28 is given as an argument to this function it will return 3, as 28_{10} is 11100_2

```
int CountOnes(unsigned int n)
{
      if (n==0)
      return 0;
      else if (n==1)
      return 1;
      else
      {
       return CountOnes(n/2)+ CountOnes(n%2);
      }

}
```

Dynamic Safe Arrays & Variants	
Question No. 3	**[Time: 15 Min] [Marks: 10]**

A two dimensional array of characters can be considered as a field. Each cell is either water 'W' or a tree 'T'. A forest is a collection of connected trees. Two trees are connected if they share a side i.e. if they are adjacent to each other. Your task is, given the information about the field, print the size of the largest forest. Size of a forest is the number of trees in it. See the sample case for clarity;

Sample Input:

5 TTTWW TWWTT TWWTT TWTTT WWTTT	First line contains the size of the matrix N. The next N lines contain N characters each, either 'W' or 'T'.

Sample Output: 10

```
void CountTreesMax(int i, int j, int N)
{
    if(i<0 || i>=N || j<0 || j>=N ||M[i][j] != 'T')
        return;
    count++;
    M[i][j]='*';
    CountTreesMax(i-1,j,N);
    CountTreesMax(i+1,j,N);
    CountTreesMax(i,j-1,N);
    CountTreesMax(i,j+1,N);
}

int main()
{
 for(int i=0;i<N;i++)
    {
        for(int j=0;j<N;j++)
        {
            if(M[i][j] == 'T')
            {
                count=0;
               CountTreesMax(i,j,N);
                if(max<count)
                    max=count;
            }
        }
    }
    cout<< max <<endl;
}
```

Linked List and Variants	
Question No. 4	**[Time: 25 Min] [Marks: 15]**

a. Given a SinglyLinkedList, write a function that remove all duplicate values from the list. Your function should handle the case that if two adjacent nodes are the same, the function should remove the second appearance of the duplicate value. [5]

```
void RemoveDuplicates(Node<T> * head)
{
  Node<T> * curr,*temp,*duplicate;
  curr=head;

 while(curr != 0 && curr->Next !=0)
 {
   temp=curr;

      while(temp->Next != 0)
       {
          if (curr->Data == temp->Next->Data)
           {
              duplicate=temp->Next;
              temp->Next=temp->Next->Next;
              duplicate->Next=0;
              delete(duplicate);
            }
           else
             temp=temp->Next;
        } // inner while
       curr= curr->Next;
      } // outer while
 } // end of RemoveDuplicates
```

b. Given a Circular Singly Linked List(CSLL) with only head as a pointer to access it. Write a function that insert a given node at the middle of CSLL. You need to identify the hidden cases as well. [5]

```
void InsertAtMiddle(Node<T> *rhs);
```

```
void InsertAtMiddle(Node<T> *n)
{
  Node<T> * oneStep,*twoSteps;
  oneStep=twoSteps=head;

 while(twoSteps != 0)
 {
   oneStep= oneStep->Next;
   twoSteps= twoSteps->Next->Next;

      if(twoSteps->Next == head)
      {
       n->Next= oneStep->Next;
       oneStep->Next=n;
       break;
      }
      if(twoSteps->Next->Next == head)
      {
       n->Next= oneStep->Next;
       oneStep->Next=n;
       break;
      }
  } //infinite while
```

c. Write a function for DoublyLinkedList<T> class, which decide for a given node pointer, if the node contains a pair of predecessors and successors or not? [5]

```
bool hasPairsOfPredecessorNSuccessor(DNode<T> *PPSNode);
```

```
bool hasPairsOfPredecessorNSuccessor(DNode<T> *PPSNode)
{
    bool PairOfSuccessors=0;
    bool PairOfPredecessor=0;
    if (PPSNode != 0)
    {
     if (PPSNode->Next !=0)
        if (PPSNode->Next->Next !=0)
        PairOfSuccessors=1;
     if (PPSNode->Prev !=0)
        if (PPSNode->Prev->Prev !=0)
        PairOfPredecessor=1;
    }

 return( (PPSNode !=0) && PairOfSuccessors &&
PairOfPredecessor);
}
```

<The End.>