University of Jordan **King Abdullah II School for Information Technology – Data Structures**

Mid term exam **24/3/2009**

اسم الطالب:..... اسم المدرس:.....

Student . #:_ _ Section #:_____ Seat #:_

Q1) Given the following definition of a **Singly Linked List**, use the functions whose prototype appear in the public or Private section (**if needed**) to answer questions **A-C** :

```
class SLList
{
private:
        SLLNode * head;      // points to the first node in a list
        void copyList(const  SLList  & otherList);  //copy from one list to another

public:
        int size( );                          //return number of  nodes
        void deleteNode(int el);              //delete node from linked list
        bool  Palindrome(int el);             //check number Palindrome(if you read it
                                              //from left or right it's the same) or not
};
```

A. Write the **Copy Constructor** for the above class. **(3 marks)**

```
SLList::SLList(const SLList & otherList)
{
  head = NULL;  è 1 Mark
  copyList(otherList);  è 2 Mark
}
```

B. Define a member function **sort** that will sort the elements in the above list in ascending (تصاعدي) order. **(7 marks)**

```
void SLList::sort()
{
  SLLNode  *tmp;        } 1 Mark
  int data;
  for(int i=0;i<size ()-1;i++)        è 1 Mark
  {
        tmp=head;              è 1 Mark
        for(int j=0;j<size ()-1-i;j++)     è 1 Mark
        {
              if(tmp->data>tmp->next->data)       è 1 Mark
              {
                    data=tmp->data;
                    tmp->data=tmp->next->data;     } 1 Mark
                    tmp->next->data=data;
              }
              tmp=tmp->next;  è 1 Mark
        }
   }
}
```

C. Write a definition of a member function **Count_Palindrome** that will count number of nodes that contain Palindrome numbers. **(5 marks)**

```
int SLList:: Count_Palindrome ( )
{
    SLLNode *tmp=head;
    int count=0;                                   1 Mark

    while (tmp!=NULL )  è 1 Mark
    {
        if (Palindrome (tmp->info))
        count++;                                   1 Mark

        tmp=tmp->next;  è 1 Mark

        return count;   è 1 Mark
    }
}
```

Q2) Given the following definition of a **Doubly Linked List,** use the functions whose prototype appear in the public section (**if needed**) to answer questions A,B :

```
template <class T>
class DLList
{
private:
     DLLNode<T> * head;      // pointer that points to first node

public:
    void addtotail (T el);                  // adds element at end
    void addtohead(T  el);                  // adds element at begin
};
```

A. Add a definition of a member function **printBackward** to class **Doubly Linked List** that will print all Even elements of a list Backward**.** **(5 marks)**

```
Template <class T>          è 1 Mark
Void DLList<T>::printBackward( )
{
DLLNode<T> * current=head;
While(current->next!=NULL)
current=current->next;                          1 Mark

While(current!=NULL)  è 1 Mark
{
If(current->info%2==0)    è 1 Mark
cout<< current->info;
current=current->prev;                          1 Mark
}
}
```

B. Write a member function **Merge** that will concatenate 2 **Doubly Linked List** into one. **(5 marks)**

```
template <class T>
void DLList<T>::merge(DLList  l)              1 Mark
{
   DLLNode <T>*tmp=head;     1 Mark
   for(;tmp->next!=0;tmp=tmp->next);          1 Mark
   tmp->next=l.head;                 1 Mark
   l.head->prev=tmp;                 1 Mark
}
```

Q3) Given the following definition of **Circular Doubly linked List** use the functions whose prototype appear in the public section (**if needed**) to answer questions A:

```
template <class T>
class CDLList
 {
        CDLLNode<T> * tail;        // Pointer points to the last node
public:
        bool is_empty( )       //returns true if the Circular Doubly linked List is empty
        int size( )            //return number of nodes in Circular Doubly linked List
                .
                .
                .
};
```

A) Define a member function **deleteTail** that will delete the Last node from **Circular Doubly linked List** defined above. **(5 marks)**

```
template<class T>
void CDLList<T>::deleteTail()
{
if(tail->prev==tail)
{
        delete tail;
        tail=0;
}
else
{
        CDLLNode <T>*tmp=tail;
        tail->prev->next=tail->next;    1 Mark
        tail->next->prev=tail->prev;    1 Mark
        tail=tail->prev;     1 Mark
        delete tmp;
}
```

1 Mark

1 Mark