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

Mid term exam **30/10/2008**

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 deletefromhead( );              //deletes the first node
        void deleteNode(int el);            //delete node from linked list
        bool checkPrime(int el);            //check number prime or not
};
```

A. Write the **assignment Overload operator** function for the above class. **(4 marks)**

```
const SLList & SLList::operator=(const SLList& otherList)   è 1 Mark
{
    if(this != &otherList)
            copyList(otherList);   è 2 Mark

    return *this;   è 1 Mark
}
```

B. Add a definition of function **deleteFromtail** to a class **Singly Linked List** that will delete the last node from the above defined linked list and return the value of the deleted node. **(5 marks)**

```
int SLList::deleteFromtail()
{
    int el;
    if (head->next == NULL)      è 1 Mark
    {
            el = head ->info;
            delete head;    è 1 Mark
            head = NULL;
    }
    else
    {
            SLLNode * tmp=head;        è 1 Mark
        for ( ; tmp->next-> next != NULL; tmp = tmp->next);  è 1 Mark
            el = tmp->next->info;
            delete tmp->next;
            tmp->next=NULL;
    }
    return el;}
```

1 Mark

C. Write a definition of a function **delete_Prime** that will delete all nodes that contain Prime integers. **(5 marks)**

```
void SLList::delete_Prime( )
{
    SLLNode *tmp1=head,*tmp2=head->next;          è 1 Mark
    while (tmp1!=NULL )          è 1 Mark
    {
        if (checkPrime(tmp1->info))          è 1 Mark
        {
            cout<<"deleting  the Prime:"<<tmp1->info;
            deleteNode(tmp1->info);      è 1 Mark
        }
        if(tmp2!= NULL)
        {
            tmp1=tmp2;   è 1 Mark
            tmp2=tmp2->next;
        }
        else
            tmp1=tmp2;
    }
}
```

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 DLLList
{
private:
     DLLNode<T> * P;      // pointer that points to any node in
                          //  a list(maybe first, last, or any position)

public:
     void addatpos(int pos,T el);          //adds element at position
     void addtotail (T el);                // adds element at end
     void addtohead(T  el);                // adds element at begin
};
```

A. Add a definition of a function **printForword** to class **Doubly Linked List** that will print all elements of a list Forwords**.** **(5 marks)**

```
Template <class T>          è 1 Mark
Void DLLList<T>::printForword( )
{
While(P->prev!=NULL)  è 1 Mark
P=P-> prev;

While(P!=NULL)  è 1 Mark
{
cout<< P->info;        è 1 Mark
P=P->next;       è 1 Mark
}
}
```

B. Add a definition of a function **addsorted** to a **Doubly Linked List** that will add an element in a correct position into a linked list that contains elements **sorted decreasingly**. **(6 marks)**

```
Template <class T>
Void DLList<T>::addsorted(T el)
{
for ( ;P->prev!=NULL ;P=P->prev); è 1 Mark
int p=0;  è 1 Mark
while(P!=NULL && P->info>el)  è 2 Mark
{
P=P->next;
p++;  è 1 Mark
}
Addatpos(p,el);          è 1 Mark
}
```

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

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

A) Define a member function **deletehead** that will delete the first node from **Circular Singly linked List** defined above. **(5 marks)**

```
template<class t>
void CSLList <t>:: deletehead ()
{
    if(tail = =NULL)
    cout<<"Cannot delete from an empty linked list"<<endl;   } 1 Mark
   else
     if(tail->next= =tail)                                   } 1 Mark
     {
            delete tail;
            tail=0;
     }
    else
    {
            node <t>*tmp=tail->next; 1 Mark
            tail->next= tmp->next;  1 Mark
            delete tmp; 1 Mark
    }
```