The University of Jordan	**KASIT/CS Department**
Data structures (Mid exam)	**First Semester(09/10)**
Name:	**Reg.no:**
Section: **Lecturer:**	**Seat no.:**

NOTE:
In all questions assume that the functions that appear in class definition as prototypes are fully defined, and you must use them to define your functions if needed.

Q1) Given the following definition of a list:

```
const int max=100;
class list
{
        int L[max];
        int length;
public:

                .
                .
};
```

A) What is the type of the list above **(2 marks)**

Contiguous list è 2 Mark

B) Write a member function **search_last** that will return the last position of the particular element in the list. **(4 marks)**

```
int list::serach_last(int el) è 1 Mark
{
  for(int i=length-1;i>=0;i--) è 1 Mark
       if(el==L[i])
            return i;        } 1 Mark
       return -1;    è 1 Mark
}
```

C) Write a member function **search_all** that will print positions of all occurrences of particular element from the above defined list. **(4 marks)**

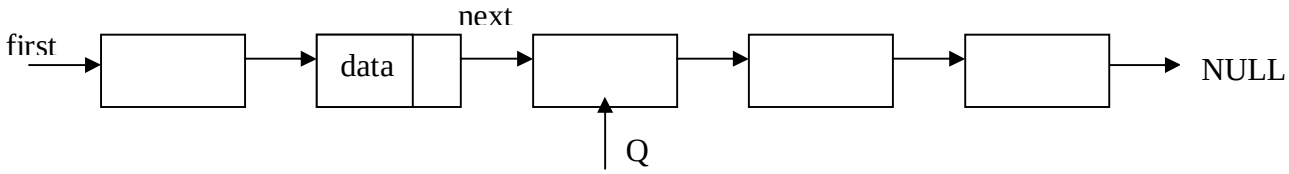
```
void list::serach_all(int el) è 1 Mark
{
     for(int i=0;i<length;i++) è 1 Mark
          if(el==L[i])  è 1 Mark
               cout<<i; è 1 Mark

}
```

Q2) Given the following list:

```
template <class t>
class list
        {
                node<t>* first;
        public:
                bool is_empty( );
                        .
        };
```

where node is

```
template <class t>
struct node
{
        t data;
        node<t>* next;
                .
                .
};
```



A) write a code that will delete the node pointed to by pointer Q from the above list. **(4 marks)**

```
node<t> *tmp=first;  è 1 Mark
  for( ;tmp->next != Q;tmp=tmp->next);     è 1 Mark
  tmp->next=Q->next;     è 1 Mark
  delete Q; è 1 Mark
```

B) Define a member function **delete_backwards** that will delete all nodes from the above list in reverse order. **(4 marks)**

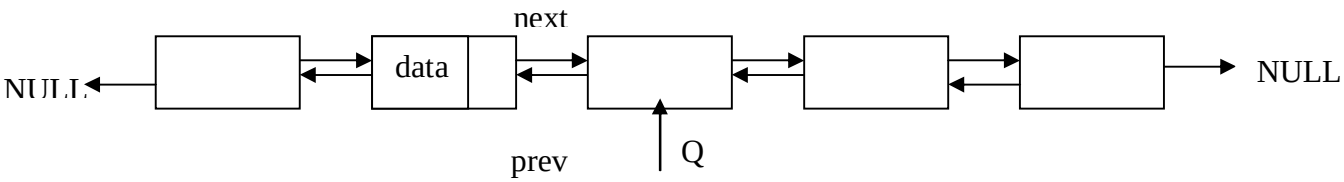
```
template<class t>
void list<t>::delete_backwards()
{

    while(!is_empty()) è 1 Mark
    {
        node<t> *tmp=first;
        while(tmp->next->next !=0)        1 Mark
            tmp=tmp->next;
        delete tmp->next;   è 1 Mark
        tmp->next=0;    è 1 Mark
    }
}
```

Q3) Given the following definition of a doubly linked list:

```
template <class t>
class list
        {
                node<t>* Q;
        public:
                        .

                        .
        };
```

where node is

```
template <class t>
struct node
{
        t data;
        node<t>* next;
        node<t>* prev;
        node( ) . . .
        node( t d,node<t>*p=0,node<t>*n=0)
        { data=d;prev=p;next=n;}
};
```



A) Write a code that will add a node with element t after a node pointed to by pointer Q.
(3 marks)

```
Q->next=Q->next->prev=new node<t>(el,Q,Q->next);
```

1 Mark 1 Mark 1 Mark

B) Write a member function **print_for( )**that will print the elements of the above list forwards.
(3 marks)

```
template<class t>
void list<t>::print_for()
{
   node<t> *tmp=Q;
     for( ;tmp->prev != 0;tmp=tmp->prev); è 1 Mark
     for( ;tmp!= 0;tmp=tmp->next)
          cout<<tmp->data;
}
```

1 Mark

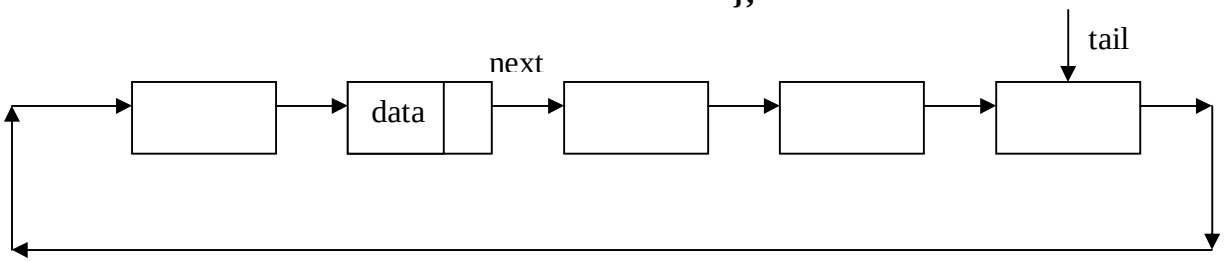
1 Mark

Q4) Given the following list:

```
template <class t>
class list
        {
                node<t>* tail;
        public:
                bool is_empty( );
                        .
        };
```

where node is

```
template <class t>
struct node
{
        t data;
        node<t>* next;
                .
                .
};
```



A) Define a proper **costructor** for a linked list defined above. **(2 marks)**

```
template<class t>
list<t>::list()
{
   tail=0;
}
```

1 Mark

è 1 Mark

A) Define a function **delete_head(t &el)** for the above class, that will delete first node. **(4 marks)**

```
template<class t>
void list<t>::delete_head(t &el)
{
   node<t> *tmp=tail->next;
   tail->next=tmp->next;
    el=tmp->data;
    delete tmp;
}
```

è 1 Mark

è 1 Mark

è 1 Mark

è 1 Mark