Data Structures and Algorithms

Lab 6

Submitted to:

Sir Rehan Ahmed

Submitted by:

Rabia batool

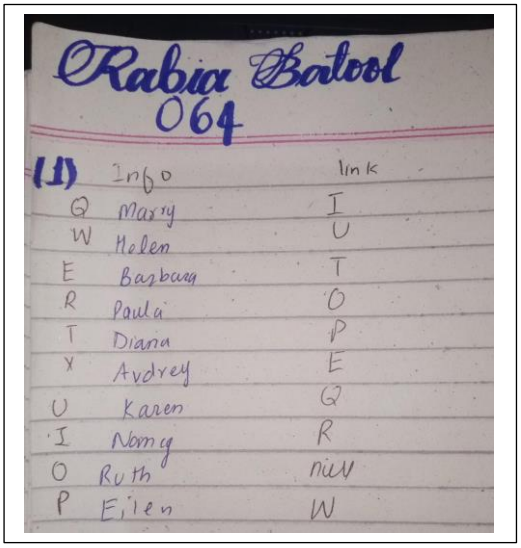
2022-BSE-064

Task 1 :

1.

The following list of names is assigned (in order) to a linear array INFO. Assign value to LINK and START, so that INFO, LINK and START form an alphabetical list.

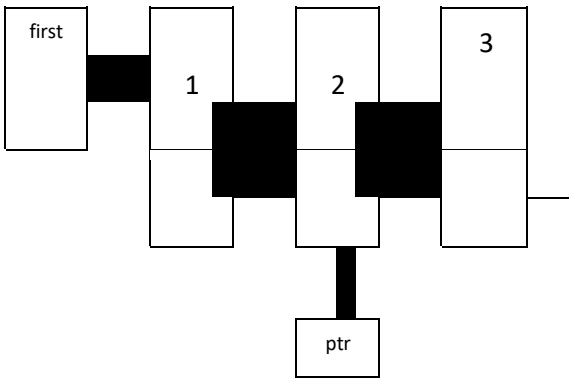
START

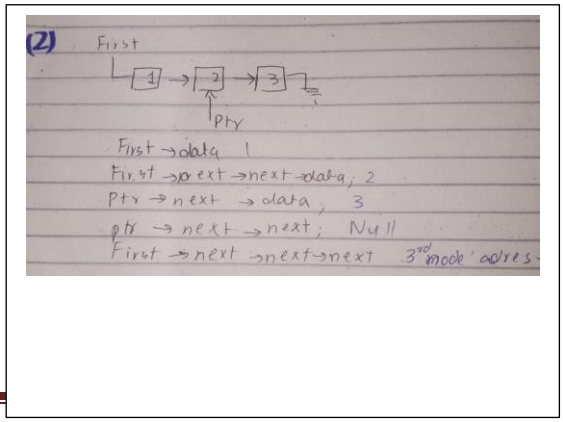


	INFO	LINK
Q	Mary	
W	Helen	
E	Barbara	
R	Paula	
T	Diana	
Y	Audrey	
U	Karen	
I	Nancy	
O	Ruth	
P	Eileen	

2.

Given the following linked list, state what does each of the following statements refer to.

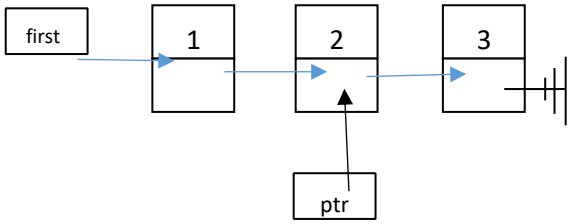




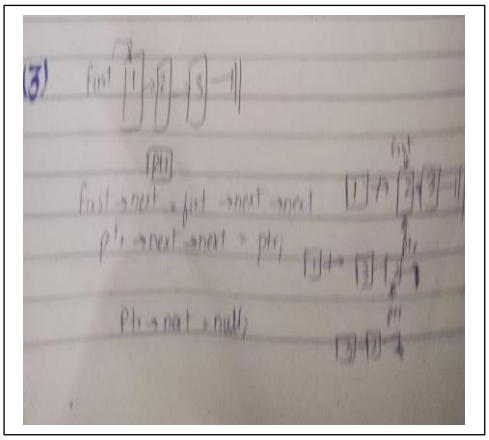
`first->data;`	`1`
`first->next->next->data;`	`2`
`ptr->next->data;`	`3`
`ptr->next->next;`	`null`
`first->next->next->next;`	`3 node address`

3.

Redraw the following list after the given instructions are executed:




```
first -> next = first -> next -> next;
ptr -> next -> next = ptr;
ptr->next = NULL;
```



Task 2 :
Implement the following exercises.

Exercise 1

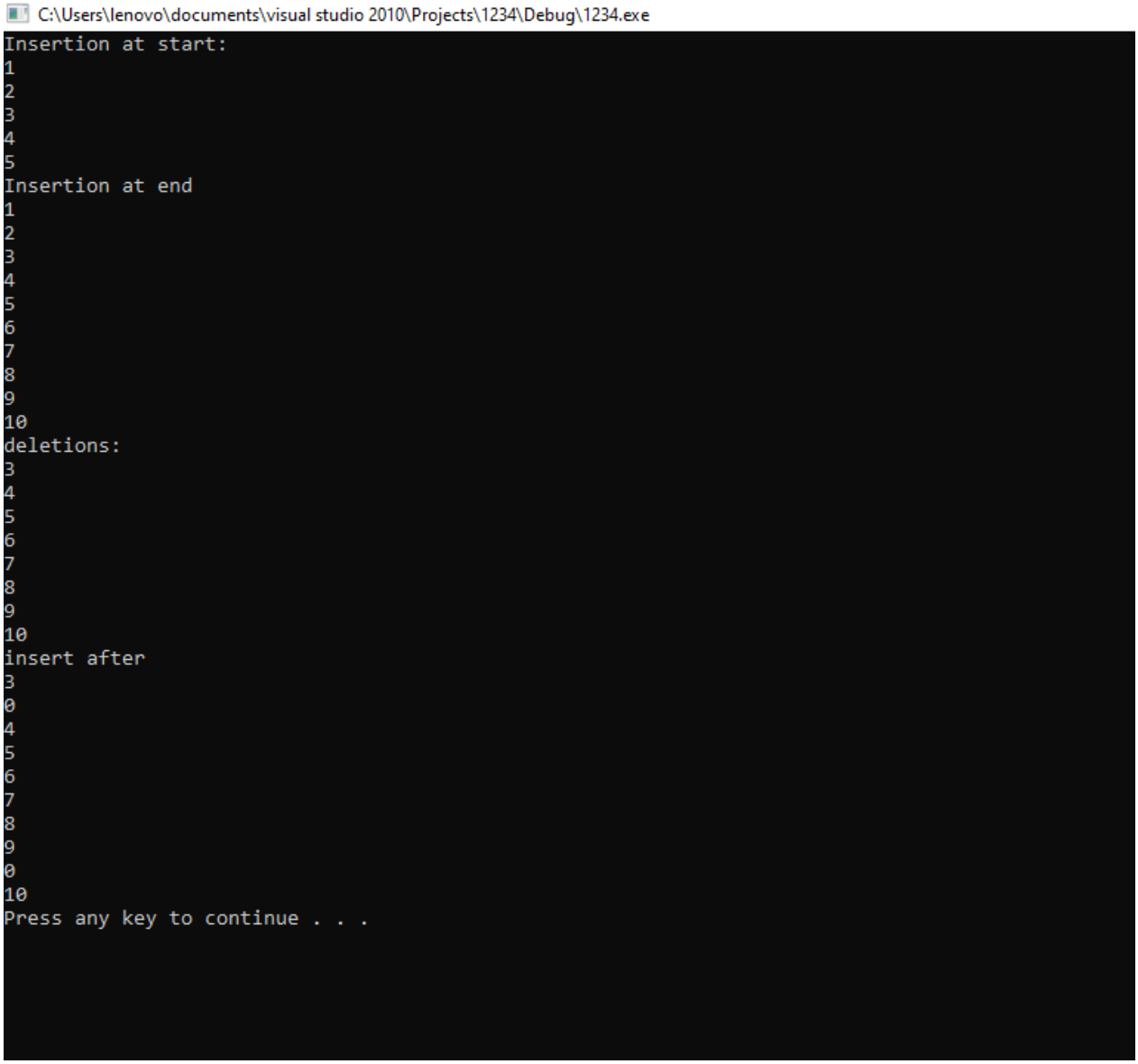
Implement the class Linked List to create a list of integers. You need to provide the implementation of the member functions as described in the following.

```
class List
{ private:
struct node
      {int data;
node *next;
      }   *head;

public:
      List();
      ~List();

      bool emptyList();// Checks if the list is empty or not

      void insertafter(int oldV, int newV);
 // Inserts a new node with value 'newV' after the node  containing
value 'oldV'. If a node with value 'oldV' does  not exist, inserts
the new node at the end.



      void deleteNode(int value);
     // Deletes the node containing the specified value

      void insert_begin(int value);
     // Inserts a new node at the start of the list

      void insert_end(int value);
     // Inserts a new node at the end of the list

      void traverse();
    // Displays the values stored in the list
};
```

```
// 1234.cpp : Defines the entry point for the console application.
//

#include "stdafx.h"
#include<iostream>
using namespace std;


class List
{
private:
struct node
{int data;
node *next;
} *head;
public:
List()
{head=NULL;
}

bool emptyList()
{if(head==NULL)
return true;
else
       return false;}
void insertafter(int oldV, int newV)
{ node* temp = new node;
    temp->data = newV;
    temp->next = NULL;

    if (head == NULL) {
        head = temp;
       } else if(oldV<head->data)
       {node*ptr=new node;
ptr->data=newV;
ptr->next=NULL;
ptr->next=head;
head=ptr;
       }
       else
       {node *ptr;
     ptr = head;
while (ptr->data != oldV)
{ptr = ptr->next;
}
temp -> next = ptr -> next;
ptr -> next = temp;  }
}


void deleteNode(int value)
{int flag=0;
node *s1,*s2,*temp;
       if(head==NULL)
{cout<<"linklist is empty"<<endl;}
else if(head->data=value)
```

```
{node *temp=head;
head=head->next;
temp->next=NULL;
delete temp;
       }
else
{
s1=head;
s2=s1->next;
while(s2->next!=NULL)
{if(s2->data==value)
{temp=s2;
s2=temp->next;
s1->next=s2;
temp->next=NULL;
delete temp;
flag++;}
else
{s1=s1->next;
s2=s2->next;}
}
if(flag==0)
{temp=s1->next;
s1->next=NULL;
delete temp;}}




}
void insert_begin(int value)

{node*ptr=new node;
ptr->data=value;
ptr->next=NULL;
ptr->next=head;
head=ptr;

}
void insert_end(int value)
{
    node* temp = new node;
    temp->data = value;
    temp->next = NULL;

    if (head == NULL) {
        head = temp;
    } else {
        node* s = head;
        while (s->next != NULL) {
            s = s->next;
        }
        s->next = temp;
    }
}
```

```
void traverse()
{node *ptr=head;
	while (ptr!=NULL)
	{cout<<ptr->data<<endl;
		ptr=ptr->next;
	}
}

};


int _tmain(int argc, _TCHAR* argv[])
{

List l;
cout<<"Insertion at start:"<<endl;
l.insert_begin(5);
l.insert_begin(4);
l.insert_begin(3);
l.insert_begin(2);
l.insert_begin(1);
l.traverse();
cout<<"Insertion at end"<<endl;
l.insert_end(6);
l.insert_end(7);
l.insert_end(8);
l.insert_end(9);
l.insert_end(10);
l.traverse();
cout<<"deletions:"<<endl;
l.deleteNode(1);
l.deleteNode(2);
l.traverse();
cout<<"insert after"<<endl;
l.insertafter(3,0);
l.insertafter(9,0);
l.traverse();
	system("pause");
	return 0;
}
```


C:\Users\lenovo\documents\visual studio 2010\Projects\1234\Debug\1234.exe
Insertion at start:
1
2
3
4
5
Insertion at end
1
2
3
4
5
6
7
8
9
10
deletions:
3
4
5
6
7
8
9
10
insert after
3
0
4
5
6
7
8
9
0
10
Press any key to continue . . .