NATIONAL UNIVERSITY OF COMPUTER AND EMERGING SCIENCES

PROGRAM: SOFTWARE ENGINEERING

DATA STRUCTURES LAB

LAB TASK-04

SUBMITTED BY:

Name: Ahmed Ali

Roll No: 22P-9318

INSTRUCTOR NAME: Sir Saood Sarwar

A DEPARTMENT OF COMPUTER SCIENCE

Q1 CODE:

```
#include<iostream>

using namespace std;


class node

{

        private:

                int data;

                node *next;


        public:

                node(int data)

                {

                        this->data=data;

                        this->next=nullptr;

                }


                friend class SLL;

};


class SLL

{

        private:

                node *head;


        public:

                SLL()

                {
```

```
        head=nullptr;
}

void insert(int data)
{
        node *newnode=new node(data);

        if(head==nullptr)
        {
                head=newnode;
        }
        else
        {
                node *temp=head;
                while(temp->next!=nullptr)
                {
                        temp=temp->next;
                }
                temp->next=newnode;
        }
}

void insert_at_pos(int val, int pos)
{
        if(pos<0)
        {
                cout<<"incorrect position entered"<<endl;
                return;
        }
        node *newnode= new node(val);

        if(pos==0)
        {
                newnode->next=head;
                head=newnode;
        }
        else
        {
                node *temp=head;
                for(int i=0; i<pos-1 && temp!=nullptr; i++)
                {
                        temp=temp->next;
                }
                if(temp==nullptr)
                {
                        cout<<"incorrect position entered"<<endl;
                        delete newnode;
                        return;
                }
                newnode->next=temp->next;
                temp->next=newnode;
        }
}

void remove()
{
        if(head==nullptr)
        {
                cout<<"Empty list"<<endl;
                return;
        }

        if(head->next==nullptr)
        {
                delete head;
                head=nullptr;
        }
        else
        {
                node *temp=head;
                while(temp->next->next!=nullptr)
                {
                        temp=temp->next;
                }
                delete temp->next;
                temp->next=nullptr;
        }
}

void remove_elem(int val)
{
        if(head==nullptr)
        {
                cout<<"Empty list"<<endl;
                return;
        }
        if(head->data==val)
        {
                node *temp=head;
                head=head->next;
                delete temp;
                return;
        }
        node *temp=head;

        while(temp->next!=nullptr && temp->next->data!=val)
        {
                temp = temp->next;
        }
        if(temp->next==nullptr)
        {
                cout<<"Element not found"<<endl;
                return;
        }
        node *remov=temp->next;
        temp->next=temp->next->next;
        delete remov;
}



void remove_by_pos(int pos)
{
        if(pos<0)
        {
                cout<<"incorrect position entered"<<endl;
                return;
        }
        if(head==nullptr)
        {
                cout<<"Empty list"<<endl;
                return;
        }
        if(pos==0)
        {
                node *temp=head;
                head=head->next;
                delete temp;
        }
        else
        {
                node *temp=head;
                for(int i=0; i<pos-1 && temp!=nullptr; i++)
                {
                        temp=temp->next;
                }
                if(temp==nullptr || temp->next!=nullptr)
                {
                        cout<<"incorrect position entered"<<endl;
                        return;
                }
                node *remov=temp->next;
                temp->next=temp->next->next;
                delete remov;
        }
```

```
}



void display()
{
	node *temp=head;
	while(temp!=nullptr)
	{
		cout<<temp->data<<" "<<endl;
		temp=temp->next;
	}
	cout<<endl;
}

int length()
{
	int count=0;
	node *temp=head;
	while(temp!=nullptr)
	{
		count++;
		temp=temp->next;
	}
	return count;
}

int search(int val)
{
	node *temp=head;
```

```
        int indx=0;
        while(temp!=nullptr)
        {
                if(temp->data==val)
                {
                        return indx;
                }
                temp=temp->next;
                indx++;
        }
        return -1;
}

void update(int pos, int new_value)
{
        if(pos<0)
        {
                cout<<"incorrect position entered"<<endl;
                return;
        }

        node *temp=head;
        for(int i=0; i<pos && temp!=nullptr; i++)
        {
                temp=temp->next;
        }

        if(temp==nullptr)
        {
```

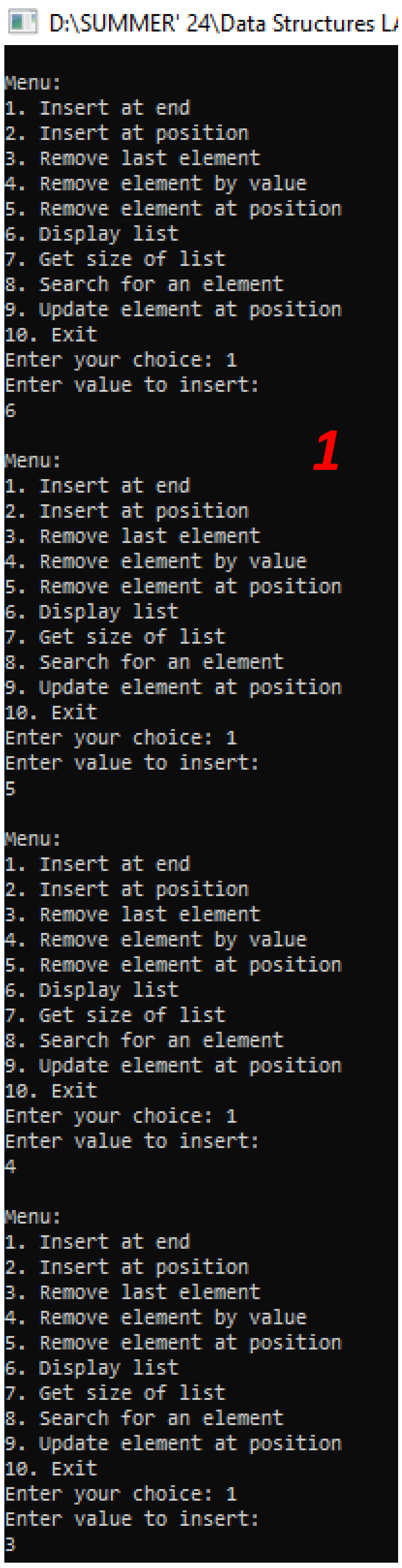
```
                        cout<<"incorrect position entered"<<endl;
                        return;
                    }
                    temp->data=new_value;
                }
};

int main()
{
        SLL list;

        int ch, value, position;
        do
        {
                cout<<endl<<"Menu:"<<endl;
                cout<<"1. Insert at end"<<endl;
                cout<<"2. Insert at position"<<endl;
                cout<<"3. Remove last element"<<endl;
                cout<<"4. Remove element by value"<<endl;
                cout<<"5. Remove element at position"<<endl;
                cout<<"6. Display list"<<endl;
                cout<<"7. Get size of list"<<endl;
                cout<<"8. Search for an element"<<endl;
                cout<<"9. Update element at position"<<endl;
                cout<<"10. Exit"<<endl;
                cout<<"Enter your choice: ";
                cin>>ch;

                switch(ch)
```

```
{
        case 1:
                cout<<"Enter value to insert: "<<endl;
                cin>>value;
                list.insert(value);
                break;

        case 2:
                cout<<"Enter value to insert: "<<endl;
                cin>>value;
                cout<<"Enter position to insert: "<<endl;
                cin>>position;
                list.insert_at_pos(value, position);
                break;

        case 3:
                list.remove();
                break;

        case 4:
                cout<<"Enter value to remove: "<<endl;
                cin>>value;
                list.remove_elem(value);
                break;

        case 5:
                cout<<"Enter position to remove: "<<endl;
                cin>>position;
                list.remove_by_pos(position);
                break;

        case 6:
                list.display();
                break;

        case 7:
                cout<<"Size of list: "<<list.length()<<endl;
                break;

        case 8:
                cout<<"Enter value to search: "<<endl;
                cin>>value;
                position=list.search(value);
                if(position!=-1)
                {
                        cout<<"Element found at position: "<<position<<endl;
                }
                else
                {
                        cout<<"Element not found"<<endl;
                }
                break;

        case 9:
                cout<<"Enter position to update: "<<endl;
                cin>>position;
                cout<<"Enter new value: "<<endl;
                cin>>value;
```

```
                        list.update(position, value);
                        break;

                case 10:
                        cout<<"Exiting............"<<endl;
                        break;

                default:
                        cout<<"incorrect choice, try again"<<endl;
        }
    }
    while(ch!=10);
  return 0;
}
```

Output-01:

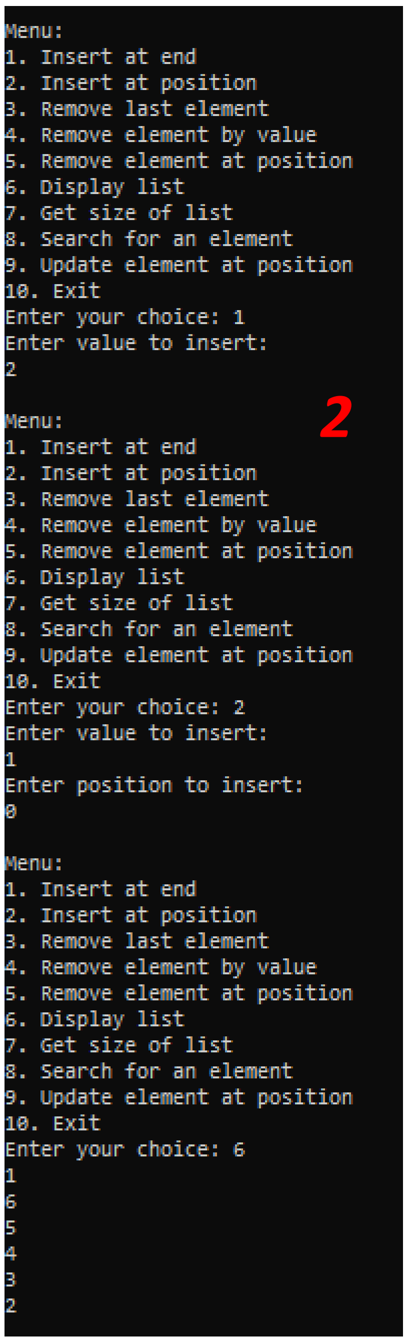
SEE BELOW (Next Page)



1

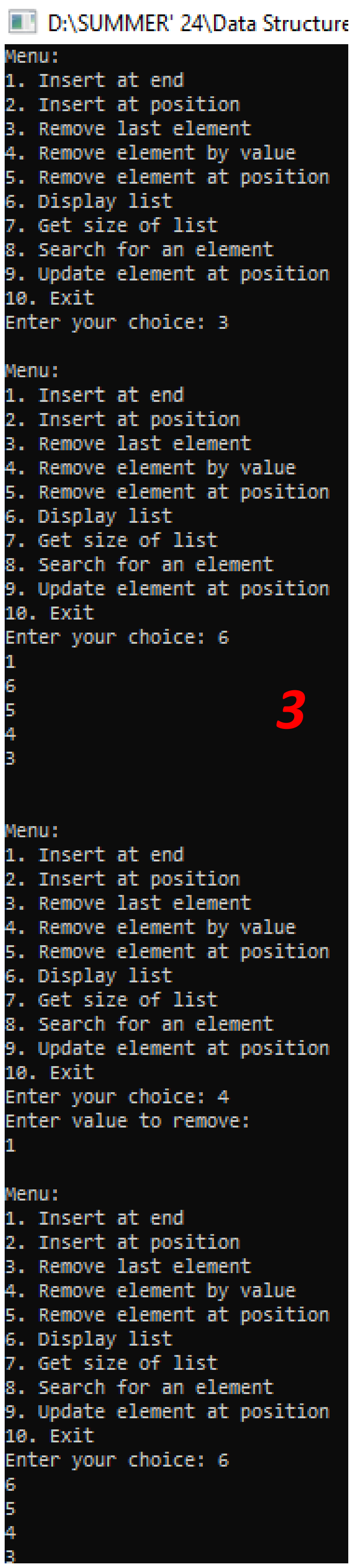
D:\SUMMER' 24\Data Structures L

```
Menu:
1. Insert at end
2. Insert at position
3. Remove last element
4. Remove element by value
5. Remove element at position
6. Display list
7. Get size of list
8. Search for an element
9. Update element at position
10. Exit
Enter your choice: 1
Enter value to insert:
6

Menu:
1. Insert at end
2. Insert at position
3. Remove last element
4. Remove element by value
5. Remove element at position
6. Display list
7. Get size of list
8. Search for an element
9. Update element at position
10. Exit
Enter your choice: 1
Enter value to insert:
5

Menu:
1. Insert at end
2. Insert at position
3. Remove last element
4. Remove element by value
5. Remove element at position
6. Display list
7. Get size of list
8. Search for an element
9. Update element at position
10. Exit
Enter your choice: 1
Enter value to insert:
4

Menu:
1. Insert at end
2. Insert at position
3. Remove last element
4. Remove element by value
5. Remove element at position
6. Display list
7. Get size of list
8. Search for an element
9. Update element at position
10. Exit
Enter your choice: 1
Enter value to insert:
3
```



2

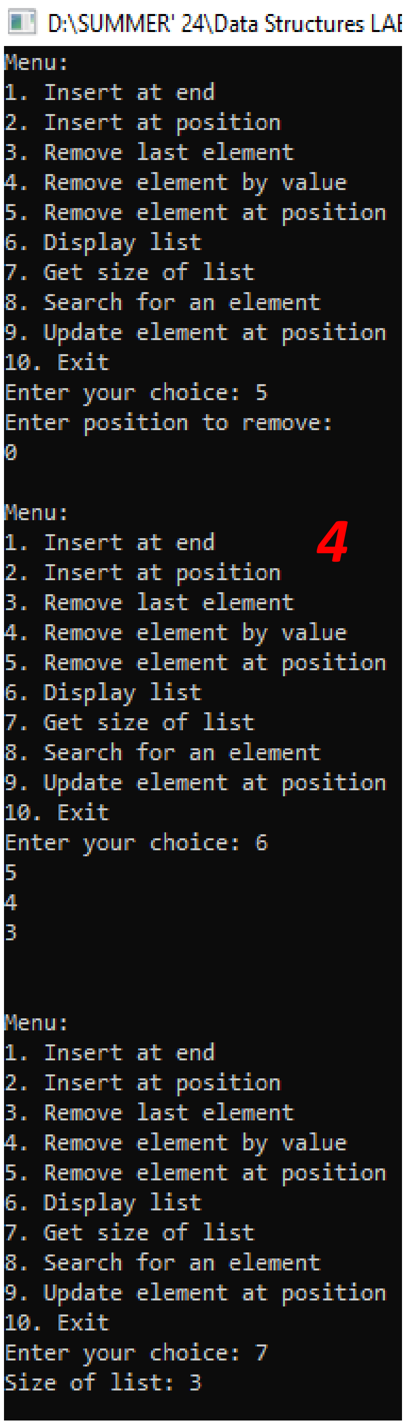
```
Menu:
1. Insert at end
2. Insert at position
3. Remove last element
4. Remove element by value
5. Remove element at position
6. Display list
7. Get size of list
8. Search for an element
9. Update element at position
10. Exit
Enter your choice: 1
Enter value to insert:
2

Menu:
1. Insert at end
2. Insert at position
3. Remove last element
4. Remove element by value
5. Remove element at position
6. Display list
7. Get size of list
8. Search for an element
9. Update element at position
10. Exit
Enter your choice: 2
Enter value to insert:
1
Enter position to insert:
0

Menu:
1. Insert at end
2. Insert at position
3. Remove last element
4. Remove element by value
5. Remove element at position
6. Display list
7. Get size of list
8. Search for an element
9. Update element at position
10. Exit
Enter your choice: 6
1
6
5
4
3
2
```



3

D:\SUMMER' 24\Data Structure

```
Menu:
1. Insert at end
2. Insert at position
3. Remove last element
4. Remove element by value
5. Remove element at position
6. Display list
7. Get size of list
8. Search for an element
9. Update element at position
10. Exit
Enter your choice: 3

Menu:
1. Insert at end
2. Insert at position
3. Remove last element
4. Remove element by value
5. Remove element at position
6. Display list
7. Get size of list
8. Search for an element
9. Update element at position
10. Exit
Enter your choice: 6
1
6
5
4
3


Menu:
1. Insert at end
2. Insert at position
3. Remove last element
4. Remove element by value
5. Remove element at position
6. Display list
7. Get size of list
8. Search for an element
9. Update element at position
10. Exit
Enter your choice: 4
Enter value to remove:
1

Menu:
1. Insert at end
2. Insert at position
3. Remove last element
4. Remove element by value
5. Remove element at position
6. Display list
7. Get size of list
8. Search for an element
9. Update element at position
10. Exit
Enter your choice: 6
6
5
4
3
```

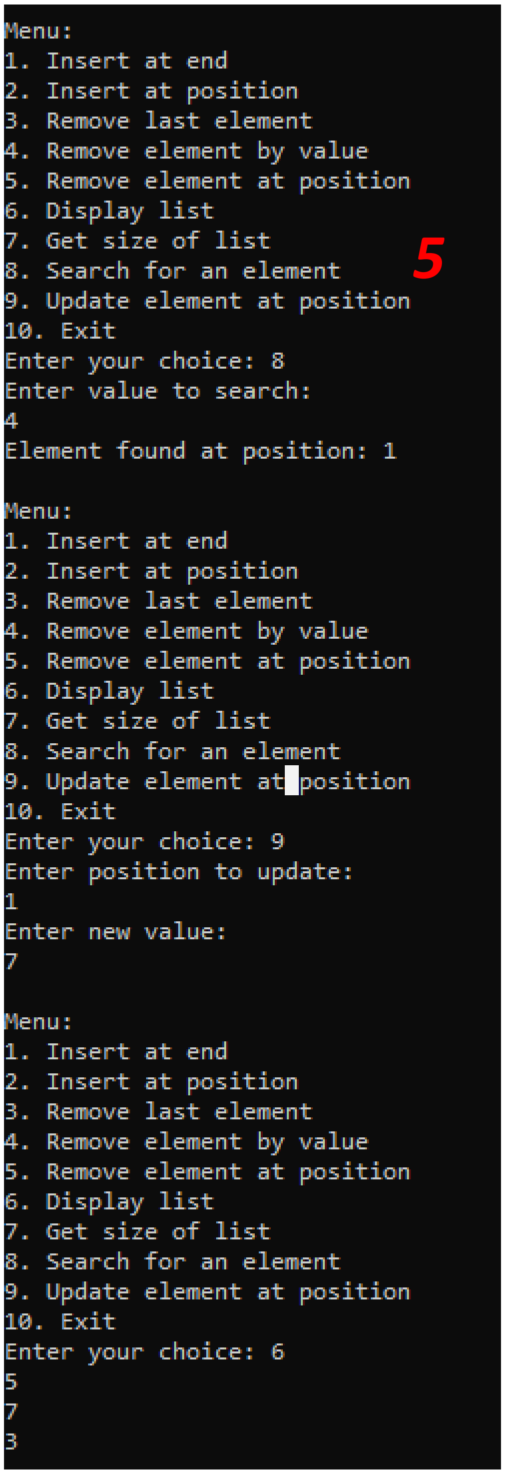


4

D:\SUMMER' 24\Data Structures LA

```
Menu:
1. Insert at end
2. Insert at position
3. Remove last element
4. Remove element by value
5. Remove element at position
6. Display list
7. Get size of list
8. Search for an element
9. Update element at position
10. Exit
Enter your choice: 5
Enter position to remove:
0

Menu:
1. Insert at end
2. Insert at position
3. Remove last element
4. Remove element by value
5. Remove element at position
6. Display list
7. Get size of list
8. Search for an element
9. Update element at position
10. Exit
Enter your choice: 6
5
4
3


Menu:
1. Insert at end
2. Insert at position
3. Remove last element
4. Remove element by value
5. Remove element at position
6. Display list
7. Get size of list
8. Search for an element
9. Update element at position
10. Exit
Enter your choice: 7
Size of list: 3
```

SEE BELOW



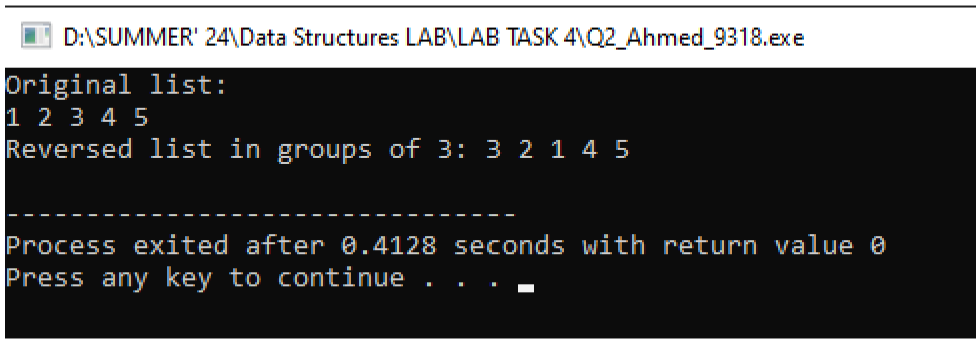
```
Menu:
1. Insert at end
2. Insert at position
3. Remove last element
4. Remove element by value
5. Remove element at position
6. Display list
7. Get size of list
8. Search for an element
9. Update element at position
10. Exit
Enter your choice: 8
Enter value to search:
4
Element found at position: 1

Menu:
1. Insert at end
2. Insert at position
3. Remove last element
4. Remove element by value
5. Remove element at position
6. Display list
7. Get size of list
8. Search for an element
9. Update element at position
10. Exit
Enter your choice: 9
Enter position to update:
1
Enter new value:
7

Menu:
1. Insert at end
2. Insert at position
3. Remove last element
4. Remove element by value
5. Remove element at position
6. Display list
7. Get size of list
8. Search for an element
9. Update element at position
10. Exit
Enter your choice: 6
5
7
3
```

5

Q2 CODE:

```
#include<iostream>
using namespace std;

class node
{
        public:
                int val;
                node *next;
                node(int val)
                {
                        this->val=val;
                        next=nullptr;
                }
};

node *reverse(node *start, node *end)  //this is a helper function
{
        node *slow=nullptr;
        node *curr=start;
        node *next=nullptr;

        while(curr!=end)
        {
                next=curr->next;
                curr->next=slow;
                slow=curr;
                curr=next;
        }
        return slow; //new head
}

node *reverseK(node *head, int k)        //main function to reverse k times all important logics are here
{
        if(head==nullptr || k==1)
        {
                return head;
        }

        int length=0;                //counting number of nodes
        node *temp=head;
        while(temp!=nullptr)
        {
                length++;
                temp=temp->next;
        }

        node *last_cell=nullptr;
        node *curr=head;
        node *new_head=nullptr;

        while(length>=k)
        {
                node *start=curr;
                node *end=curr;

                for(int i=1; i<k; i++)
                {
                        end=end->next;
                }

                node *next_start=end->next;
                node *new_front=reverse(start, end->next);

                if(new_head==nullptr)
                {
                        new_head=new_front; //Updating the new head for first part
                }
                else
                {
                        last_cell->next=new_front;
                }

                start->next=next_start;

//updating pointers
                last_cell=start;
                curr=next_start;

length=length-k;
        }

        if(new_head==nullptr)
        {
                return head;
        }
        else
        {
                return new_head;
        }
}

void print(node *head)
{
        while(head!=nullptr)
        {
                cout<<head->val;
                if(head->next)
                cout<<" ";
                head=head->next;
        }
        cout<<endl;
}

int main()
{
        node *head=new node(1);
        head->next=new node(2);
        head->next->next=new node(3);
        head->next->next->next=new node(4);
        head->next->next->next->next=new node(5);

        cout<<"Original list: "<<endl;
        print(head);
        int k=3;
        head=reverseK(head, k);

        cout<<"Reversed list in groups of "<<k<<": ";
        print(head);

        return 0;
}
```

Output-02:



D:\SUMMER' 24\Data Structures LAB\LAB TASK 4\Q2_Ahmed_9318.exe

```
Original list:
1 2 3 4 5
Reversed list in groups of 3: 3 2 1 4 5

--------------------------------
Process exited after 0.4128 seconds with return value 0
Press any key to continue . . .
```