DS Assignment 9

Ques: Implement the following operations w.r.t. doubly linked list:

1) Insertion at any position given by the user(including both the ends).

2) Deletion at any position given by the user(including both the ends).

3) Display the list.

4) Search for a specific element.

5) Find maximum and minimum element.

Source Code:

```
#include<stdio.h>

#include<stdlib.h>

struct node{

        int data;

        struct     node*     next;

        struct node* previous;

};

struct node* head, *temp;

void traverse()

{

        if(head==NULL)
```

```
        {
                printf("Doubly Linked List is Empty!");
        }
        else
        {
                temp=head;
                while(temp!=NULL)
                {
                        printf("%d " ,temp->data);
                        temp=temp->next;
                }
        }
}
int main()
{
        int choice=1,operation,n,number,count=0,pos,i,min,max;
        head=NULL;
        while(choice)
        {
                struct node* newnode= malloc(sizeof(struct node));
                printf("Enter the number to be inserted in Doubly Linked
Linked: ");
                scanf("%d", &newnode->data);
                newnode->next=NULL;
                newnode->previous=NULL;
```

```
            if(head==NULL)
            {
                head=temp=newnode;
            }
            else
            {
                temp->next=newnode;
                newnode->previous=temp;
                temp=newnode;
                count++;
            }
            printf("Enter 1 to insert more else enter 0 to
Display/Traverse: ");
            scanf("%d", &choice);
        }
        traverse();
        printf("\nEnter the choice of Operation you would like to
perform");
        printf("\n1. Traverse\n2. Insertion at Beginning\n3. Insertion at a
Specified Position");
        printf("\n4. Insertion at the end\n5. Deletion at Beginning\n6.
Deletion at a Specified Position\n");
        printf("7. Deletion at end\n8. Search for an element\n9. Find
Maximum element\n10. Find Minimum Element\n11. Exit\n");
        while(1)
        {
```

```
printf("\nEnter your choice: ");
scanf("%d", &operation);
switch(operation)
{
        case 1:
                traverse();
                break;
        case 2:
                if(head==NULL)
                {
                        printf("Doubly Linked List is Empty!");
                        return;
                }
                else
                {
                        struct node* newno= malloc(sizeof(struct
node));
                        printf("Enter the Number to be inserted:
");
                        scanf("%d", &newno->data);
                        temp=head;
                        newno->next=head;
                        temp->previous=newno;
                        head=newno;
                        count++;
                }
```

```
                traverse();
                break;
            case 3:
                if(head==NULL)
                {
                    printf("Doubly Linked List is Empty!");
                    return;
                }
                else
                {
                    struct node* newnod= malloc(sizeof(struct
node));
                    printf("Enter the Number to be inserted:
");
                    scanf("%d", &newnod->data);
                    printf("Enter the Position where the
number is to be inserted:  ");
                    scanf("%d", &pos);
                    if(pos<0 || pos>count)
                    {
                        printf("Number Invalid!");
                        return;
                    }
                    else
                    {
                        temp=head;
```

```
                        for(i=1; i<pos-1; i++)
                        {
                                temp=temp->next;
                        }
                        newnod->next=temp->next;
                        temp->next->previous=newnod;
                        newnod->previous=temp;
                        temp->next=newnod;
                        count++;
                }
                traverse();
        }
        break;
    case 4:
        if(head==NULL)
        {
                printf("Doubly Linked List is Empty!");
                return;
        }
        else
        {
                struct node* newn= malloc(sizeof(struct
node));
                printf("Enter the Number to be inserted:
");
                scanf("%d", &newn->data);
```

```
                temp=head;
                while(temp->next!=NULL)
                {
                        temp=temp->next;
                }
                newn->previous=temp->next;
                temp->next=newn;
                newn->next=NULL;
                count++;
        }
        traverse();
        break;
case 5:
        if(head==NULL)
        {
                printf("Doubly Linked List is Empty!");
                return;
        }
        else
        {
                temp=head;
                head=temp->next;
                head->previous=NULL;
                temp->next=NULL;
                free(temp);
```

```
                count--;
            }
            traverse();
            break;
        case 6:
            if(head==NULL)
            {
                printf("Doubly Linked List is Empty!");
                return;
            }
            else
            {
                printf("Enter the position of Node to be
deleted: ");
                scanf("%d", &pos);
                struct node* aage;
                aage=head;
                for(i=1; i<pos+1; i++)
                    {
                        aage=aage->next;
                    }
                temp=head;
                for(i=1; i<pos; i++)
                    {
                        temp=temp->next;
```

```
                }
            struct node* peeche;
            peeche=head;
            for(i=1; i<pos-1; i++)
                {
                    peeche=peeche->next;
                }
            aage->previous=temp->previous;
            peeche->next=temp->next;
            free(temp);
            count--;
        }
        traverse();
        break;
    case 7:
        if(head==NULL)
        {
            printf("Doubly Linked List is Empty!");
            return;
        }
        else
        {
            temp=head;
            for(i=1; i<=count; i++)
            {
```

```
                temp=temp->next;
            }
            struct node* d;
            d=head;
            for(i=1; i<=count-1; i++)
            {
                d=d->next;
            }
            d->next=NULL;
            temp->previous=NULL;
            temp->next=NULL;
            free(temp);
            count--;
        }
        traverse();
        break;
    case 8:
        if(head==NULL)
        {
            printf("Doubly Linked List is Empty!");
        }
        else
        {
            int flag=0;
```

```
                printf("Enter the Number to be Searched:
");
                scanf("%d", &number);
                temp=head;
                while(temp!=NULL)
                {
                        if(temp->data==number)
                        {
                                printf("Number found!");
                                flag=1;
                                break;
                        }
                        temp=temp->next;
                }
                if(flag=0)
                {
                        printf("Number not found!");
                }
        }
        break;
    case 9:
        if(head==NULL)
        {
                printf("Doubly Linked List is Empty!");
        }
```

```
                    else
                    {
                        temp=head;
                        temp->data=max;
                        while(temp!=NULL)
                        {
                            if(temp->data>max)
                            {
                                max=temp->data;
                            }
                            temp=temp->next;
                        }
                        printf("The Maximum element is %d",
max);
                    }
                    break;
                case 10:
                    if(head==NULL)
                    {
                        printf("Doubly Linked List is Empty!");
                    }
                    else
                    {
                        min=head->data;
                        temp=head;
```

```
                        while(temp!=NULL)
                        {
                                if(temp->data<min)
                                {
                                        min=temp->data;
                                }
                                temp=temp->next;
                        }
                        printf("The Minimum element is %d", min);
                }
                break;
            case 11:
                exit(0);
            default:
                printf("Enter a Valid Number!");
        }
    }
}
```

Output:

C:\Users\Himani\Desktop\DS\DLL.exe

```
Enter the number to be inserted in Doubly Linked Linked: 2
Enter 1 to insert more else enter 0 to Display/Traverse: 1
Enter the number to be inserted in Doubly Linked Linked: 3
Enter 1 to insert more else enter 0 to Display/Traverse: 1
Enter the number to be inserted in Doubly Linked Linked: 4
Enter 1 to insert more else enter 0 to Display/Traverse: 1
Enter the number to be inserted in Doubly Linked Linked: 5
Enter 1 to insert more else enter 0 to Display/Traverse: 1
Enter the number to be inserted in Doubly Linked Linked: 6
Enter 1 to insert more else enter 0 to Display/Traverse: 1
Enter the number to be inserted in Doubly Linked Linked: 7
Enter 1 to insert more else enter 0 to Display/Traverse: 0
2 3 4 5 6 7
Enter the choice of Operation you would like to perform
1. Traverse
2. Insertion at Beginning
3. Insertion at a Specified Position
4. Insertion at the end
5. Deletion at Beginning
6. Deletion at a Specified Position
7. Deletion at end
8. Search for an element
9. Find Maximum element
10. Find Minimum Element
11. Exit

Enter your choice: 1
2 3 4 5 6 7
Enter your choice: 10
The Minimum element is 2
Enter your choice: 2
Enter the Number to be inserted: 1
1 2 3 4 5 6 7
Enter your choice: 3
Enter the Number to be inserted: 8
Enter the Position where the number is to be inserted: 4
1 2 3 8 4 5 6 7
Enter your choice: 4
Enter the Number to be inserted: 9
1 2 3 8 4 5 6 7 9
Enter your choice: 5
2 3 8 4 5 6 7 9
Enter your choice: 6
Enter the position of Node to be deleted: 3
2 3 4 5 6 7 9
Enter your choice: 7
2 3 4 5 6 7
Enter your choice: 8
Enter the Number to be Searched: 5
Number found!
```

C:\Users\Himani\Desktop\DS\DLL.exe

```
Enter your choice: 7
2 3 4 5 6 7
Enter your choice: 8
Enter the Number to be Searched: 5
Number found!
Enter your choice: 9
The Maximum element is 7
Enter your choice: 11

--------------------------------
Process exited after 79.13 seconds with return value 0
Press any key to continue . . .
```