DATA STRUCTURES LAB RECORD

NAME : ADARSH K N

USN : 1BM19CS004

COURSE : DATA STRUCTURES LAB

DEPARTMENT : CSE

SECTION : 3A

ACADEMIC YEAR : 2020 – 21

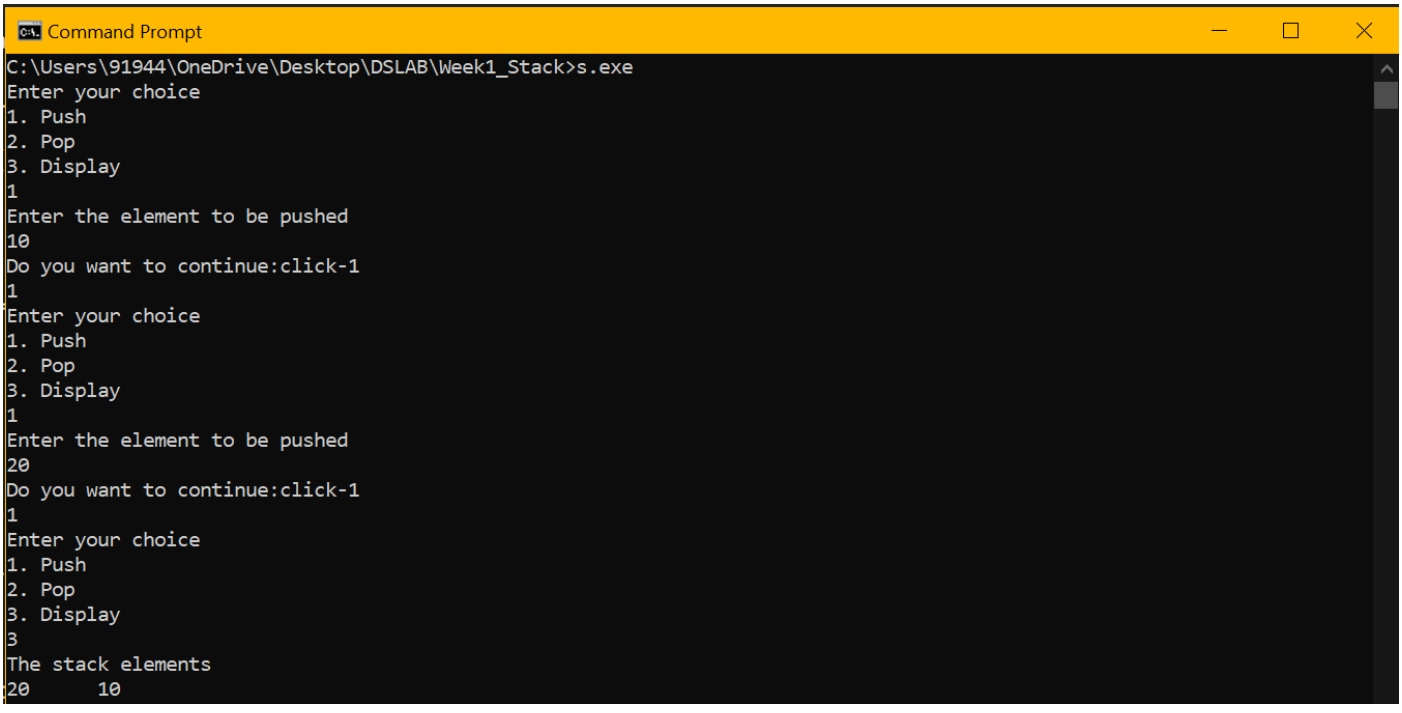
LAB PROGRAM – 1

INPUT:

```
#include <stdio.h>
#define size 5
int top=-1;
void push(int [], int);
int pop(int[]);
void display(int []);
int main()
{
   int a[5];
   int choice,element;
   int ch;
   do
   {
   printf("Enter your choice\n");
   printf("1. Push\n");
   printf("2. Pop\n");
   printf("3. Display\n");
   scanf("%d",&choice);
   switch(choice)
   {
      case 1: printf("Enter the element to be pushed \n");
            scanf("%d",&element);
            push(a,element);
            break;
      case 2: element=pop(a);
            if(element==-1)
               printf("Stack Underflow cannot remove\n");
            else
              printf("Poped element is %d \n",element);
            break;
      case 3: display(a);
            break;
      default: printf("Invalid choice\n");
   }
   printf("Do you want to continue:click-1\n");
   scanf("%d",&ch);
   } while(ch==1);
   return 0;
}

void push(int a[], int ele)
{
   if (top==size-1)
   {
      printf("Stack overflow cannot push\n");
   }
   else
```

```
    {
        top++;
        a[top]=ele;
    }
}

int pop(int a[])
{
    int ele;
    if(top==-1)

      return -1;
    else
    {
        ele=a[top];
        top--;
        return (ele);
    }
}

void display(int a[])
{
    int i;
    printf("The stack elements\n");
    for(i=top;i>=0;i--)
    {
        printf("%d\t",a[i]);
    }
    printf("\n");
}
```

OUTPUT:



LAB PROGRAM – 2

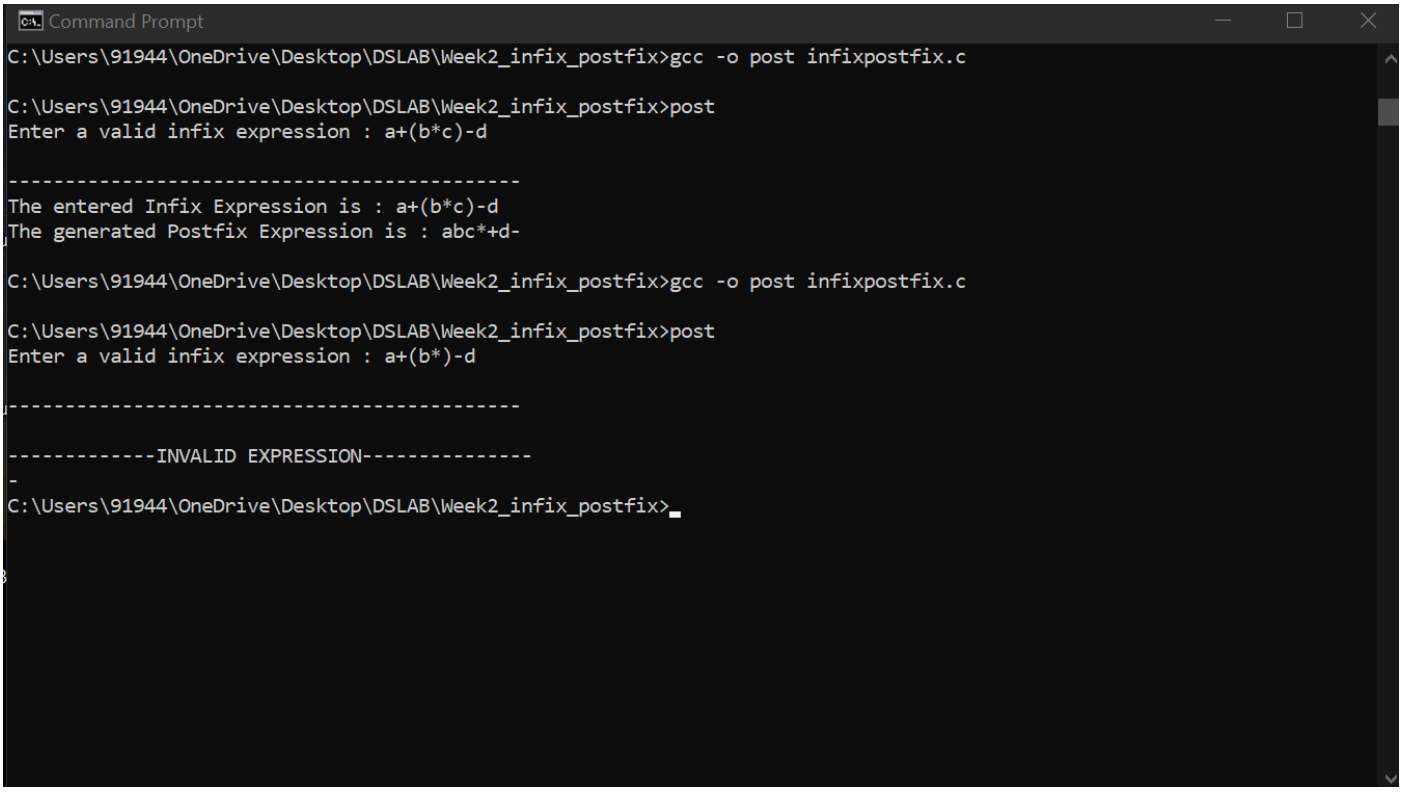
INPUT:

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#define SIZE 100


char stack[SIZE];
int top=-1;

void push(char ch)
{
   if (top==SIZE-1)
      printf("STACK OVERFLOW!! Stack is full!!\n");
   else
   {
      top++;
      stack[top]=ch;
   }
}

char pop()
{
   char item;
   if (top==-1)
      printf("\nSTACK UNDERFLOW!! Stack is empty!!");
   else
   {
      item=stack[top];
      top--;
      return item;
   }
}

int stackempty()
{
   if(top==-1)
      return 1;
   else
      return 0;
}

char stacktop()
{
   if( top==-1)
      printf("\nSTACK UNDERFLOW!! Stack is empty!!");
   else
      return stack[top];
```

```
}

int priority(char ch)
{
   switch(ch)
   {
      case '+':
      case '-':return (1);
      case '*':
      case '/': return (2);
      case '^': return (3);
      default : return (0);
   }
}

int main(int argc, char **argv)
{
   char infix[100];
   int i, item;
   printf("Enter a valid infix expression : ");
   scanf("%s",infix);
   printf("\n-----------------------------------------\n");
   for (int i = 0; i < strlen(infix); ++i)
   {
      if((infix[i]=='*' || infix[i]=='+' || infix[i]=='/' ||
         infix[i]=='-' || infix[i]=='^' || infix[i]=='(') &&
         (infix[i+1]=='*' || infix[i+1]=='+' || infix[i+1]=='/' ||
         infix[i+1]=='-' || infix[i+1]=='^' || infix[i+1]==')'))
      {
         printf("\n----------INVALID EXPRESSION----------\n-");
         exit(1);
      }
   }
   printf("The entered Infix Expression is : %s",infix);
   printf("\nThe generated Postfix Expression is : ");
   i=0;
    while (infix[i]!='\0')
   {
      switch (infix[i])
      {
         case '(': push(infix[i]);
                   break;
         case ')': while(( item=pop())!='(')
                     printf("%c",item);
                   break;
         case '+':
         case '-':
         case '*':
         case '/':
         case '^':
                   while(!stackempty() && priority(infix[i])<=priority(stacktop()))
                   {
```

```
                item=pop();

                printf("%c", item);
            }

            push(infix[i]);
            break;
        default : printf("%c", infix[i]);
            break;
    }
     i++;
  }

  while(!stackempty())
  {
    char item;
    item=pop();
    printf("%c", item);

  }
  printf("\n");
  return 0;
}
```

OUTPUT:



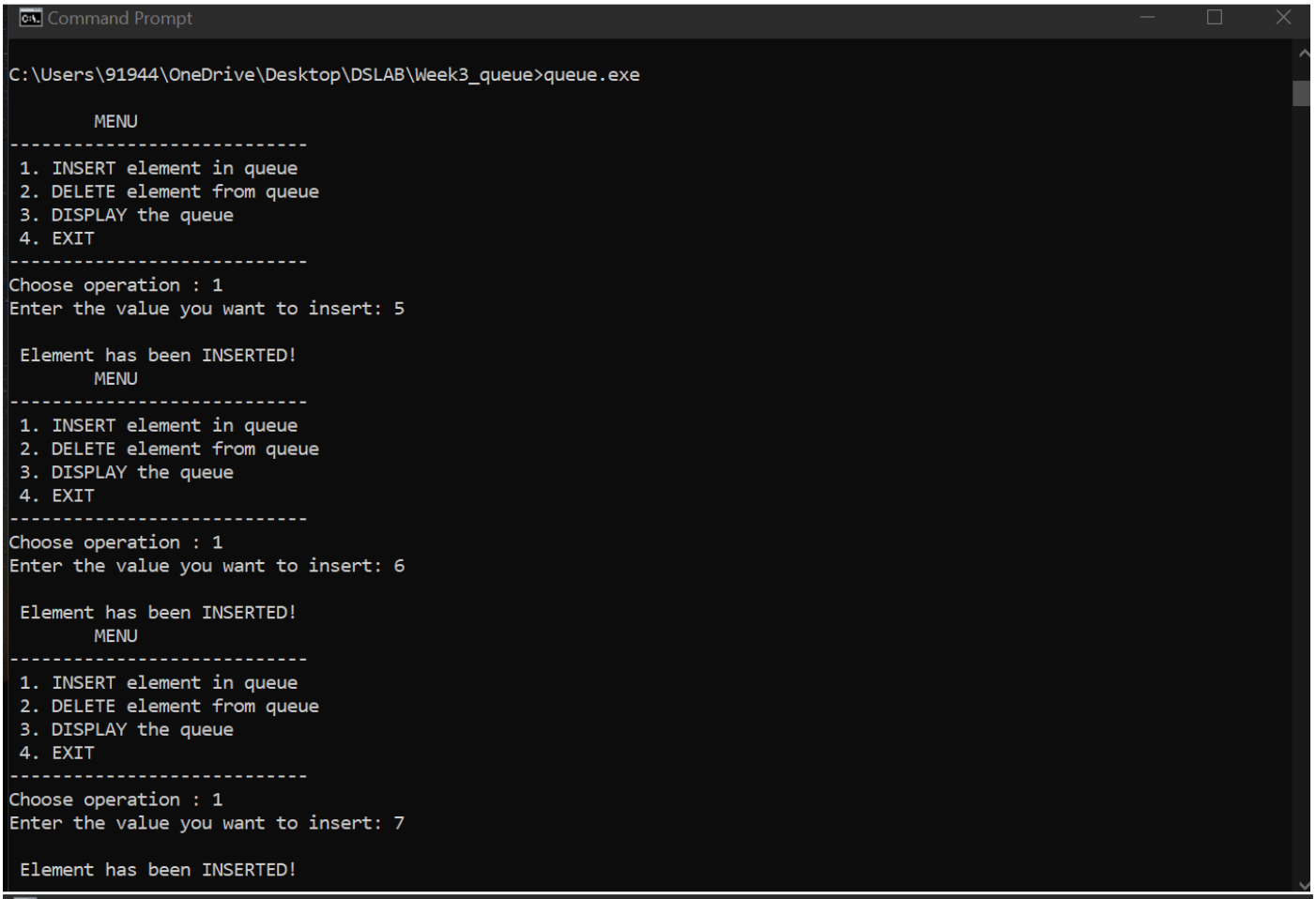
LAB PROGRAM – 3

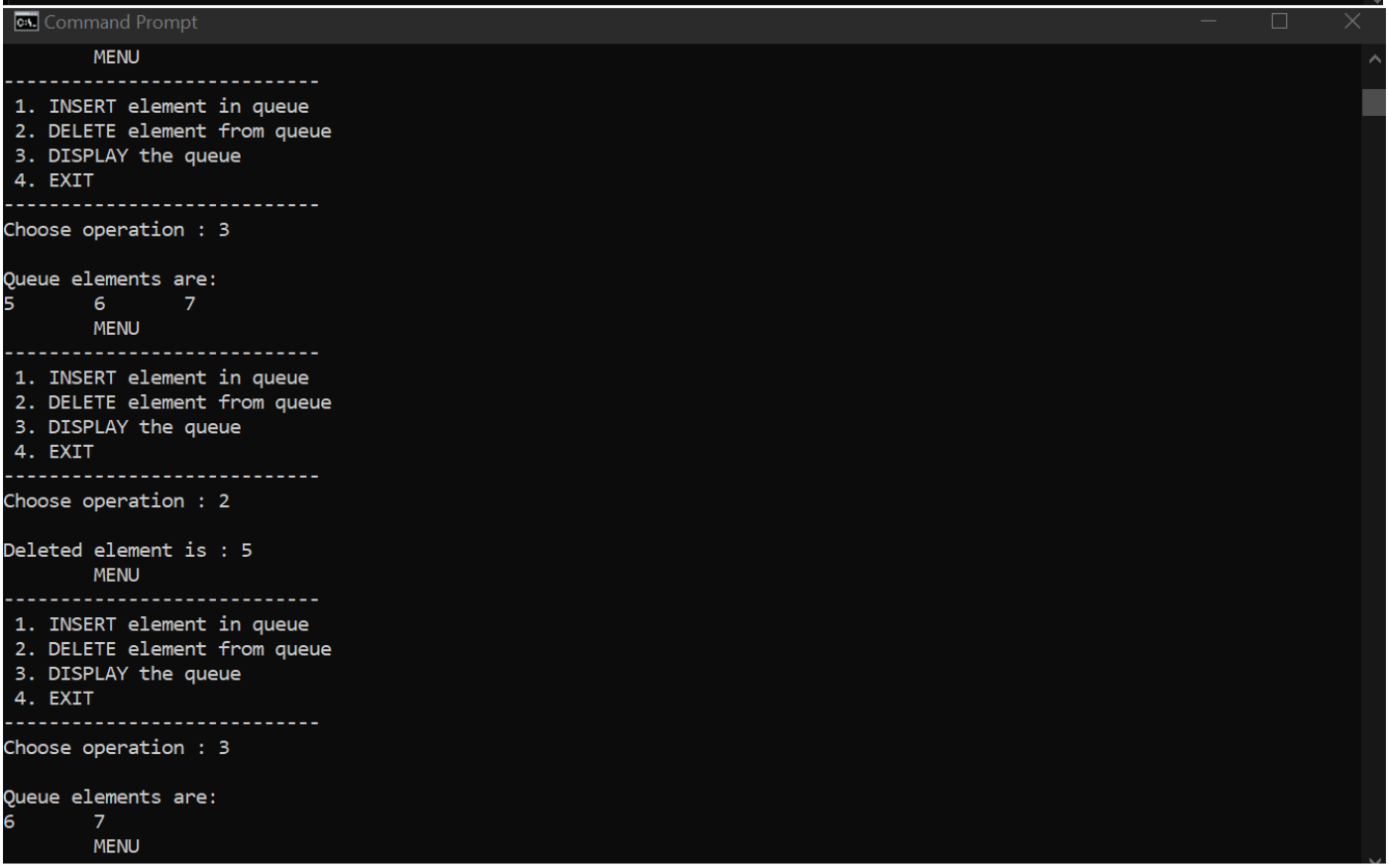
INPUT:

```
#include <stdio.h>
#include <stdlib.h>

#define N  5

int queue[N];

void enQueue(int value, int *front, int *rear)
{
  if(*rear == N-1)
    printf("\nQueue is FULL!");
  else{
    if(*front == -1)
  *front = 0;
    (*rear)++;
    queue[*rear] = value;
    printf("\n Element has been INSERTED!");
  }
}
void deQueue(int *front, int *rear)
{
  if(*rear == *front && *front == -1)
 {
   printf("\nQueue is EMPTY!\n");

 }
 else if(*front == *rear)
 {
  printf("\nDeleted element is : %d", queue[*front]);
  *front = *rear = -1;
  }
 else
 {
  printf("\nDeleted element is : %d", queue[*front]);
  *front += 1;
 }
}
void display(int *front, int *rear)
{
  if(*rear == -1)
    printf("\nQueue is EMPTY!!!");
  else{
    int i;
    printf("\nQueue elements are:\n");
    for(i = *front; i <= *rear; i++)
   printf("%d\t",queue[i]);
  }
}
```

```
int main()
{

  int value, choice;
  int front = -1, rear = -1;
  while(1)
  {
      printf("\n\tMENU");
      printf("\n-------------------------");
      printf("\n 1. INSERT element in queue");
      printf("\n 2. DELETE element from queue");
      printf("\n 3. DISPLAY the queue");
      printf("\n 4. EXIT");
      printf("\n-------------------------");
      printf("\nChoose operation : ");

      scanf("%d",&choice);
      switch(choice)
      {
         case 1:
           printf("Enter the value you want to insert: ");
           scanf("%d",&value);
           enQueue(value, &front, &rear);
           break;
         case 2:
           deQueue(&front, &rear);
           break;
         case 3:
           display(&front, &rear);
           break;
         case 4:
           exit(0);
         default: printf("\nEnter the choice CORRECTLY!!");
                  getch();
      }
   }
  return 0;
}
```

OUTPUT:


Command Prompt
C:\Users\91944\OneDrive\Desktop\DSLAB\Week3_queue>queue.exe
MENU
1. INSERT element in queue
2. DELETE element from queue
3. DISPLAY the queue
4. EXIT
Choose operation : 1
Enter the value you want to insert: 5
Element has been INSERTED!
MENU
1. INSERT element in queue
2. DELETE element from queue
3. DISPLAY the queue
4. EXIT
Choose operation : 1
Enter the value you want to insert: 6
Element has been INSERTED!
MENU
1. INSERT element in queue
2. DELETE element from queue
3. DISPLAY the queue
4. EXIT
Choose operation : 1
Enter the value you want to insert: 7
Element has been INSERTED!


Command Prompt
MENU
1. INSERT element in queue
2. DELETE element from queue
3. DISPLAY the queue
4. EXIT
Choose operation : 3
Queue elements are:
5 6 7
MENU
1. INSERT element in queue
2. DELETE element from queue
3. DISPLAY the queue
4. EXIT
Choose operation : 2
Deleted element is : 5
MENU
1. INSERT element in queue
2. DELETE element from queue
3. DISPLAY the queue
4. EXIT
Choose operation : 3
Queue elements are:
6 7
MENU

LAB PROGRAM – 4

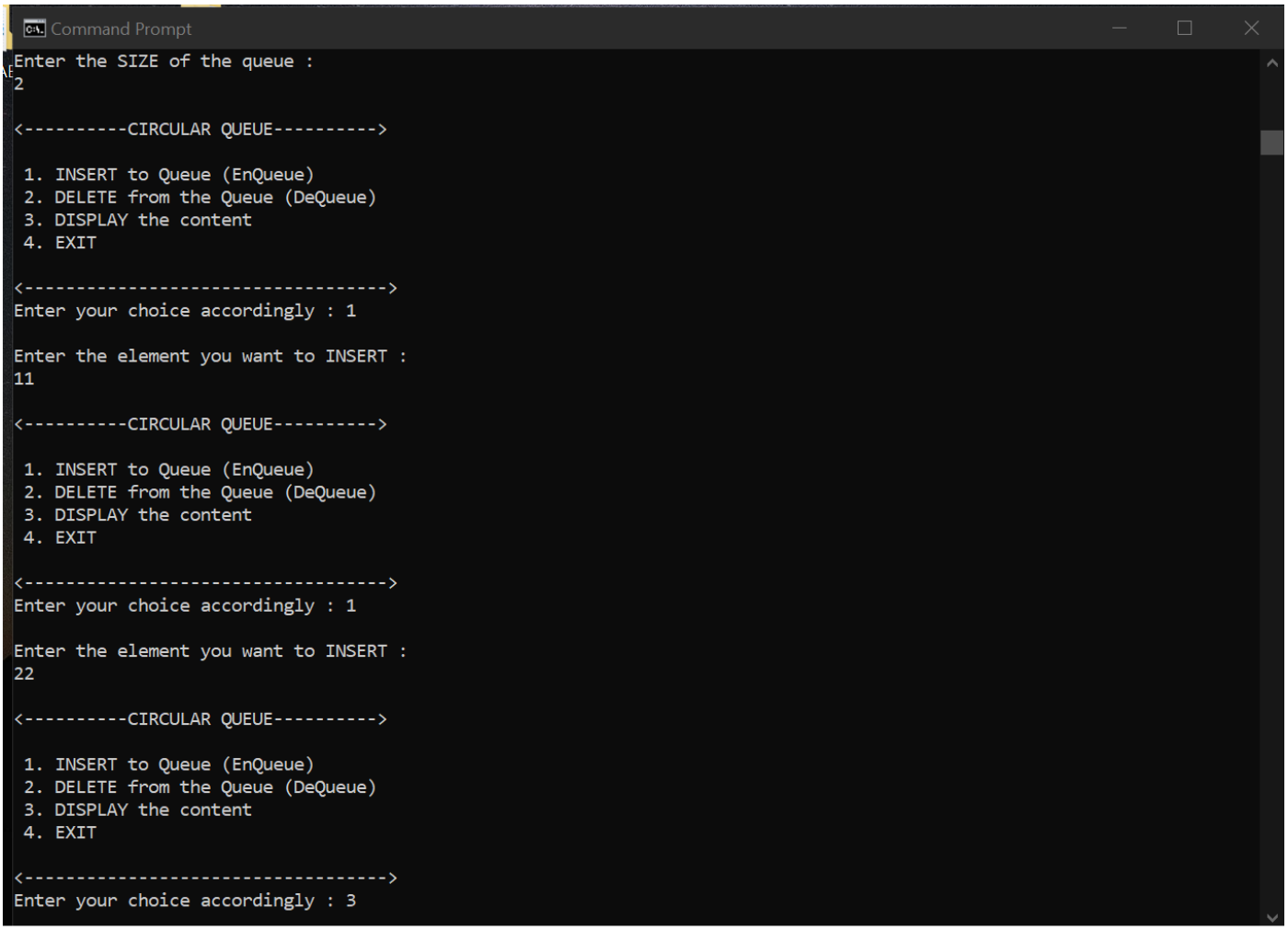
INPUT:

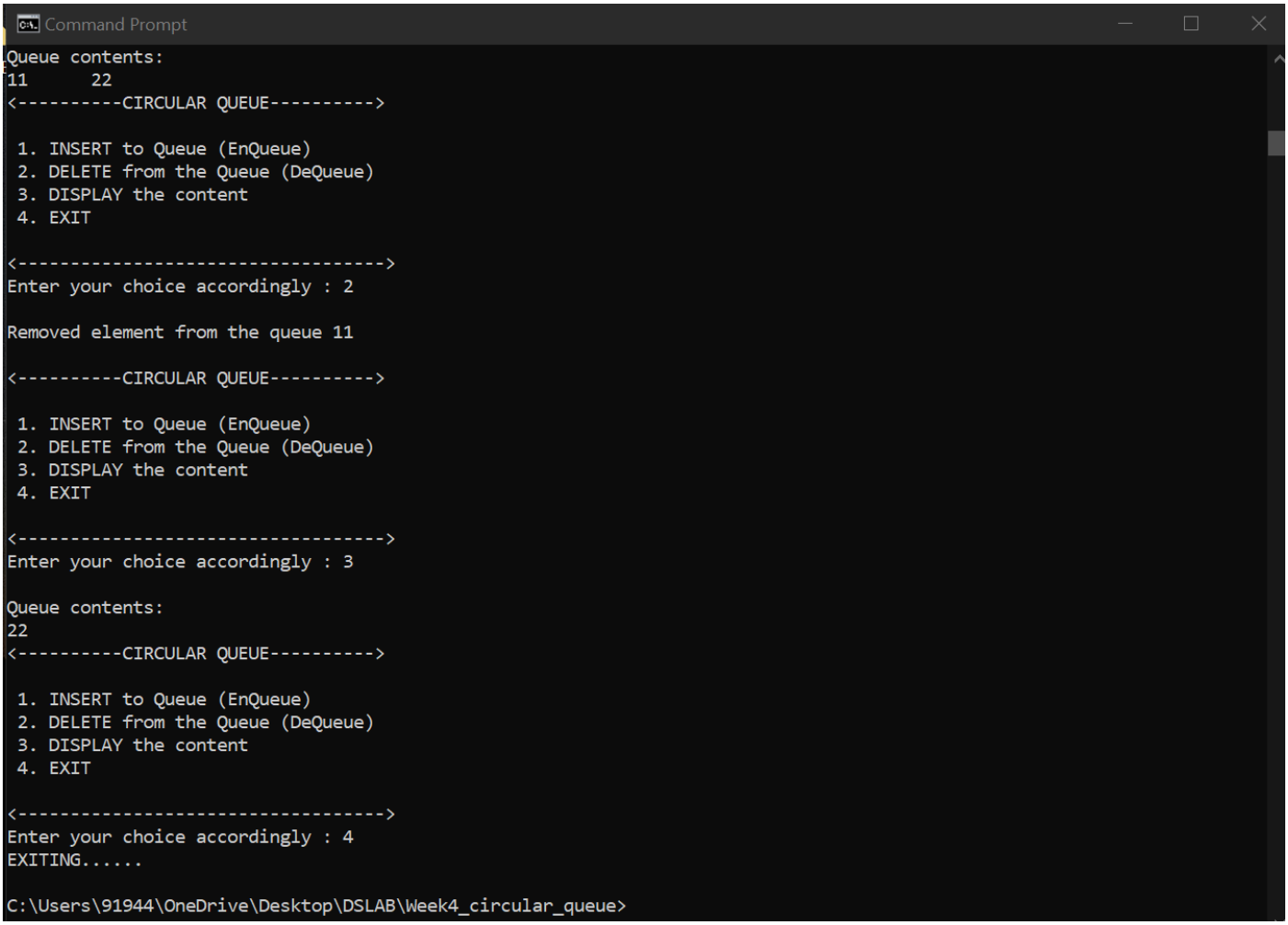
```
#include <stdio.h>
#include <stdlib.h>
#define S 3

int front=-1;
int rear=-1;

int queue[S];

void Enque(int, int);
int Deque(int);
void display(int);
int main(int argc, char **argv)
{
        int choice, SIZE;
   int item;
   printf("Enter the SIZE of the queue : \n");
   scanf("%d",&SIZE);
   do
   {
      printf("\n<---------CIRCULAR QUEUE--------->\n");
      printf("\n 1. INSERT to Queue (EnQueue)");
      printf("\n 2. DELETE from the Queue (DeQueue)");
      printf("\n 3. DISPLAY the content ");
      printf("\n 4. EXIT\n");
      printf("\n<------------------------------>\n");
      printf("Enter your choice accordingly : ");
      scanf("%d", &choice);
      switch(choice)
      {
         case 1:  if(((front == 0 && rear == SIZE - 1)) || (front == rear + 1))
                  {
                      printf("-----Queue is FULL-----\n");
                      break;
                  }

                  printf("\nEnter the element you want to INSERT : \n");
                  scanf("%d",&item);
                  Enque(SIZE, item);
                  break;

         case 2: item=Deque(SIZE);
                 if(item==-999)
                    printf("-----Queue is EMPTY-----\n");
                 else
                 printf("\nRemoved element from the queue %d\n",item);
                 break;
```

```
        case 3: display(SIZE);
                break;

        case 4: printf("EXITING.....\n");
                exit(0);
        default: printf("INVALID CHOICE!!");
                 break;
      }
   }
   while (choice!=4);
        return 0;
}

void Enque(int SIZE, int ele)
{
   if(((front == 0 && rear == SIZE - 1)) || (front == rear + 1) )
   {
     printf("-----Queue is FULL-----\n");
     return;
   }
   else
   {
    rear=(rear+1)%SIZE;
    queue[rear]=ele;
    if(front == -1)
       front=0;
   }
}

int Deque(int SIZE)
{
   int item;
   if((front == -1)&&(rear == -1))
   {
      return(-999);
   }
   else
   {
      item=queue[front];

      if(front==rear)
      {
         front=-1;
         rear=-1;
      }
      else
      {
         front=(front+1)%SIZE;
      }
      return item;
   }
```

```
}

void display(int SIZE)
{
   int i;
   if(((front==-1)&& (rear==-1)))
   {
      printf("-----Queue is EMPTY-----\n");
      return;
   }
   else
   {
      printf("\nQueue contents:\n");
      for(i=front;i!=rear;i=(i+1)%SIZE)
      {
         printf("%d\t", queue[i]);
      }
      printf("%d\t", queue[i]);
   }
}
```

OUTPUT:





LAB PROGRAM – 5

INPUT:

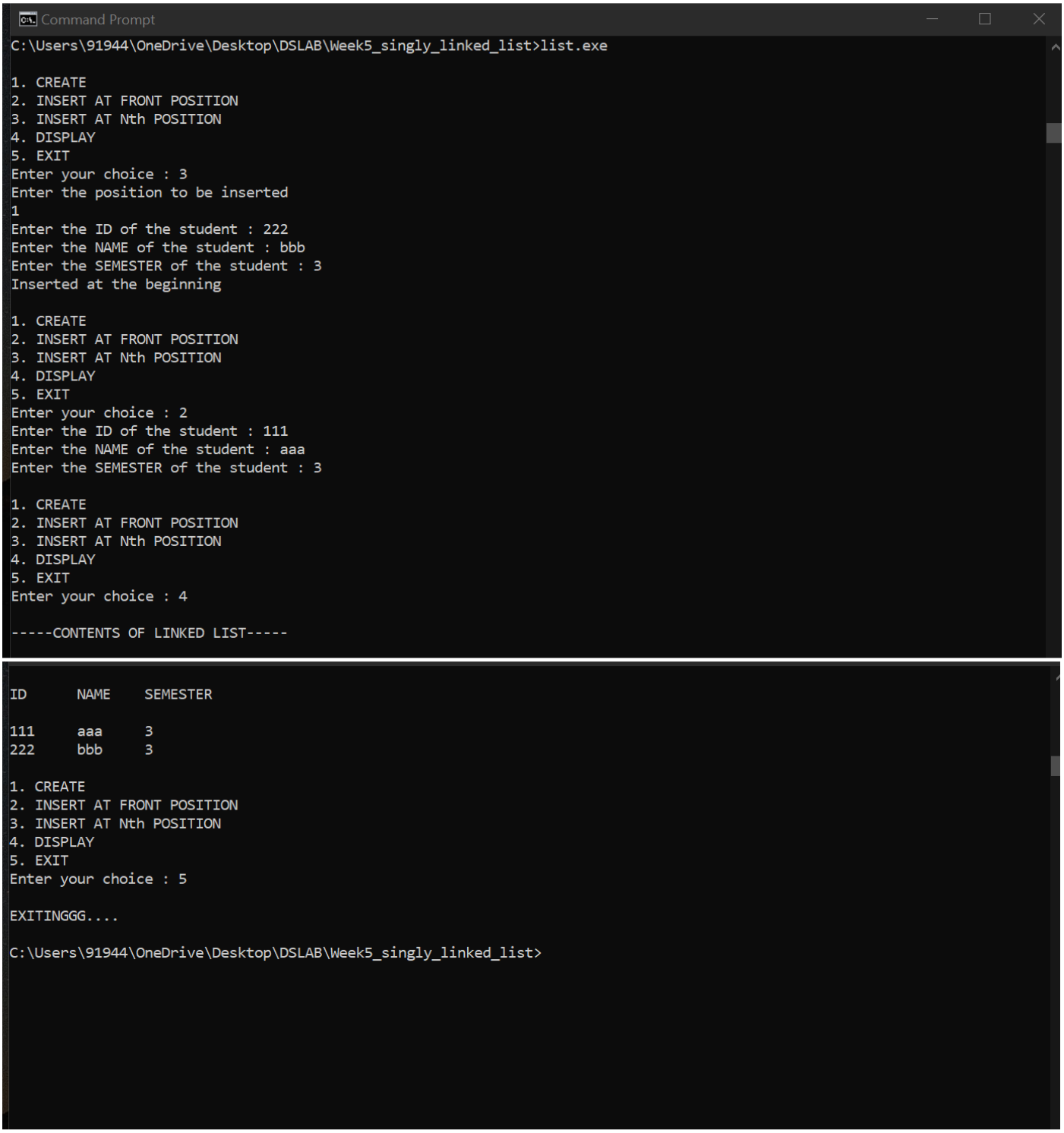
```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

void create();
void display();
void delete(int);
void insert_before();
void insert_at_nth();
struct node
{
   int id;
   char name[20];
   int sem;
   struct node *next;
};
struct node *head=NULL;
int count=0;

int main()
```

```
{
   int choice,ele;
   do
   {
   printf("\n1. CREATE \n2. INSERT AT FRONT POSITION \n3. INSERT AT Nth POSITION  \n4.
DISPLAY  \n5. EXIT");
   printf("\nEnter your choice : ");
   scanf("%d",&choice);
   switch(choice)
   {
      case 1: create();
            break;
      case 2: insert_before();
            break;
       case 3: printf("Enter the position to be inserted\n");
            scanf("%d",&ele);
            insert_at_nth(ele);
            break;
      case 4: display();
            break;
      case 5: printf("\nEXITINGGG....\n");
            break;
   }
   }while(choice != 5);
   return 0;
}

void create()
{
  struct node *newnode,*temp;
   int stu_id;
   char stu_name[20];
   int semester;
   newnode =(struct node *) malloc (sizeof(struct node));
   printf("Enter the ID of the student : ");
   scanf("%d", &stu_id);
   printf("Enter the NAME of the student : ");
   scanf("%s",stu_name);
   printf("Enter the SEMESTER of the student : ");
   scanf("%d", &semester);
   newnode->id=stu_id;
   strcpy(newnode->name, stu_name);
   newnode->sem=semester;
   if (head==NULL)
    {
      newnode->next=NULL;
     head=newnode;
     printf("Node created\n");
    }
    else
   {
   temp=head;
```

```
      while(temp->next!=NULL)
      {
            temp=temp->next;
       }
     temp->next=newnode;
     newnode->next=NULL;
      printf("Node created\n");
  }
}

void display()
{
   struct node *ptr=NULL;
   ptr=head;

   if(ptr==NULL)
   {
      printf("Nothing to print\n");
   }
   else
   {
      printf("\n-----CONTENTS OF LINKED LIST-----\n");
      printf("\nID\tNAME\tSEMESTER\n\n");
      while(ptr!=NULL)
     {
      printf("%d\t%s\t%d\n",ptr->id,ptr->name,ptr->sem);
      ptr=ptr->next;
     }
   }

}

void insert_before()
{
   struct node *newnode;
   int stu_id;
   char stu_name[20];
   int semester;
   newnode=(struct node*)malloc(sizeof(struct node));
   printf("Enter the ID of the student : ");
   scanf("%d", &stu_id);
   printf("Enter the NAME of the student : ");
   scanf("%s",stu_name);
   printf("Enter the SEMESTER of the student : ");
   scanf("%d", &semester);
   newnode->id=stu_id;
   strcpy(newnode->name, stu_name);
   newnode->sem=semester;
   newnode->next=head;
   head=newnode;
 }
```

```
 void insert_at_nth(int p)
{
   struct node *newnode;
   int stu_id;
   char stu_name[20];
   int semester;
   if(count+1<p)
      printf("The position exceeds the number of nodes");
   else
   {
   printf("Enter the ID of the student : ");
   scanf("%d", &stu_id);
   printf("Enter the NAME of the student : ");
   scanf("%s",stu_name);
   printf("Enter the SEMESTER of the student : ");
   scanf("%d", &semester);
   newnode =(struct node *) malloc (sizeof(struct node));

   newnode->id=stu_id;
   strcpy(newnode->name, stu_name);
   newnode->sem=semester;

   if(head==NULL&& p>1)
   {
     printf("List empty, enter in first position\n");
   }

   if(p==1)
   {
      printf("Inserted at the beginning\n");
      newnode->next=head;
      head=newnode;
      count++;
   }

   else
   {
      int i;
      struct node *temp1;
      temp1=head;
      for(i=2;i<p;i++)
      {
      temp1= temp1->next;
      }
      temp1->next=newnode;

      printf("Node inserted at %d position in linked list\n",p);
      count++;
      }
   }
}
```

OUTPUT:



```
Command Prompt
C:\Users\91944\OneDrive\Desktop\DSLAB\Week5_singly_linked_list>list.exe

1. CREATE
2. INSERT AT FRONT POSITION
3. INSERT AT Nth POSITION
4. DISPLAY
5. EXIT
Enter your choice : 3
Enter the position to be inserted
1
Enter the ID of the student : 222
Enter the NAME of the student : bbb
Enter the SEMESTER of the student : 3
Inserted at the beginning

1. CREATE
2. INSERT AT FRONT POSITION
3. INSERT AT Nth POSITION
4. DISPLAY
5. EXIT
Enter your choice : 2
Enter the ID of the student : 111
Enter the NAME of the student : aaa
Enter the SEMESTER of the student : 3

1. CREATE
2. INSERT AT FRONT POSITION
3. INSERT AT Nth POSITION
4. DISPLAY
5. EXIT
Enter your choice : 4

-----CONTENTS OF LINKED LIST-----

ID      NAME    SEMESTER

111     aaa     3
222     bbb     3

1. CREATE
2. INSERT AT FRONT POSITION
3. INSERT AT Nth POSITION
4. DISPLAY
5. EXIT
Enter your choice : 5

EXITINGGG....

C:\Users\91944\OneDrive\Desktop\DSLAB\Week5_singly_linked_list>
```

LAB PROGRAM – 6

INPUT:

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

void create();
void display();
void delete(int);
void delete_first();
void delete_last();
struct node
{
   int id;
   char name[20];
   int sem;
   struct node *next;
};
struct node *head=NULL;
int count=0;

int main()
{
   int choice,ele;
   do
   {
   printf("\n1. CREATE \n2. DELETE FIRST NODE \n3. DELETE Nth NODE \n4. DELETE LAST NODE
\n5. DISPLAY  \n6. EXIT");
   printf("\nEnter your choice : ");
   scanf("%d",&choice);
   switch(choice)
   {
      case 1: create();
            break;
      case 2: delete_first();
            break;
      case 3: printf("Enter the ID to be deleted\n");
            scanf("%d",&ele);
            delete(ele);
            break;
      case 4: delete_last();
            break;
      case 5: display();
            break;
      case 6: printf("\nEXITINGGG....\n");
            break;
   }
   }while(choice != 6);
   return 0;
}
```

```
void create()
{
  struct node *newnode,*temp;
   int stu_id;
   char stu_name[20];
   int semester;
   newnode =(struct node *) malloc (sizeof(struct node));
   printf("Enter the ID of the student : ");
   scanf("%d", &stu_id);
   printf("Enter the NAME of the student : ");
   scanf("%s",stu_name);
   printf("Enter the SEMESTER of the student : ");
   scanf("%d", &semester);
   newnode->id=stu_id;
   strcpy(newnode->name, stu_name);
   newnode->sem=semester;
   if (head==NULL)
    {
      newnode->next=NULL;
     head=newnode;
     printf("Node created\n");
    }
    else
   {
   temp=head;
      while(temp->next!=NULL)
      {
             temp=temp->next;
       }
     temp->next=newnode;
     newnode->next=NULL;
      printf("Node created\n");
  }
}

void display()
{
   struct node *ptr=NULL;
   ptr=head;

   if(ptr==NULL)
   {
      printf("EMPTY LIST!! Nothing to print\n");
   }
   else
   {
      printf("\n-----CONTENTS OF LINKED LIST-----\n");
      printf("\nID\tNAME\tSEMESTER\n\n");
      while(ptr!=NULL)
     {
      printf("%d\t%s\t%d\n",ptr->id,ptr->name,ptr->sem);
      ptr=ptr->next;
```

```
      }
    }
}

void delete(int ele)
{
   struct node *temp,*del=NULL;

    if (head == NULL)
    {
       printf("Empty List. Can't delete\n");
       return;
    }
    temp=head;
    if(head->id==ele)
    {
       head=head->next;
       printf("Successfully deleted the ID from the list.\n");
       return;
    }
   while (temp->next!=NULL)
   {
      if(temp->next->id==ele)
      {
         del=temp->next;
        if(del->next==NULL)
        temp->next=NULL;
        else
         temp->next=del->next;
      }
      else
        temp=temp->next;
   }
   if(del==NULL)
   {
      printf("Student ID not found in the list\n");
      return;
   }
   printf("Successfully deleted the ID from the list.\n");
}

void delete_first()
{
   struct node *toDelete;

   if(head == NULL)
   {
      printf("Empty List. Can't delete.\n");
   }
   else
   {
      toDelete = head;
```

```
        head = head->next;

        printf("ID deleted = %d\n", toDelete->id);

        free(toDelete);

        printf("Successfully deleted first node from the list.\n");
    }
}

void delete_last()
{
    struct node *toDelete, *secondLastNode;

    if(head == NULL)
    {
        printf("Empty List. Can't delete.\n");
    }
    else
    {
        toDelete = head;
        secondLastNode = head;
        /* Traverse to the last node of the list */
        while(toDelete->next != NULL)
        {
            secondLastNode = toDelete;
            toDelete = toDelete->next;
        }
        if(toDelete == head)
        {
            head = NULL;
        }
        else
        {
            /* Disconnect link of second last node with last node */
            secondLastNode->next = NULL;
        }
        /* Delete the last node */
        free(toDelete);

        printf("Successfully deleted last node from the list.\n");
    }
}
```

OUTPUT:

```
Command Prompt - list.exe
C:\Users\91944\OneDrive\Desktop\DSLAB\Week6_singly_linked_list_operations>gcc dellist.c -o list

C:\Users\91944\OneDrive\Desktop\DSLAB\Week6_singly_linked_list_operations>list.exe

1. CREATE
2. DELETE FIRST NODE
3. DELETE Nth NODE
4. DELETE LAST NODE
5. DISPLAY
6. EXIT
Enter your choice : 1
Enter the ID of the student : 111
Enter the NAME of the student : aaa
Enter the SEMESTER of the student : 3
Node created

1. CREATE
2. DELETE FIRST NODE
3. DELETE Nth NODE
4. DELETE LAST NODE
5. DISPLAY
6. EXIT
Enter your choice : 1
Enter the ID of the student : 222
Enter the NAME of the student : bbb
Enter the SEMESTER of the student : 3
Node created

1. CREATE
2. DELETE FIRST NODE
3. DELETE Nth NODE
4. DELETE LAST NODE
5. DISPLAY
6. EXIT
Enter your choice : 1
Enter the ID of the student : 333
Enter the NAME of the student : ccc
Enter the SEMESTER of the student : 3
Node created

1. CREATE
2. DELETE FIRST NODE
3. DELETE Nth NODE
4. DELETE LAST NODE
5. DISPLAY
6. EXIT
Enter your choice : 3
Enter the ID to be deleted
555
Student ID not found in the list

1. CREATE
2. DELETE FIRST NODE
3. DELETE Nth NODE
4. DELETE LAST NODE
5. DISPLAY
6. EXIT
Enter your choice : 3
Enter the ID to be deleted
222
Successfully deleted the ID from the list.

1. CREATE
2. DELETE FIRST NODE
3. DELETE Nth NODE
4. DELETE LAST NODE
5. DISPLAY
6. EXIT
Enter your choice : 5
```

```
Command Prompt

-----CONTENTS OF LINKED LIST-----

ID      NAME    SEMESTER

111     aaa     3
333     ccc     3

1. CREATE
2. DELETE FIRST NODE
3. DELETE Nth NODE
4. DELETE LAST NODE
5. DISPLAY
6. EXIT
Enter your choice : 6

EXITINGGG....

C:\Users\91944\OneDrive\Desktop\DSLAB\Week6_singly_linked_list_operations>
```

LAB PROGRAM – 7

INPUT:

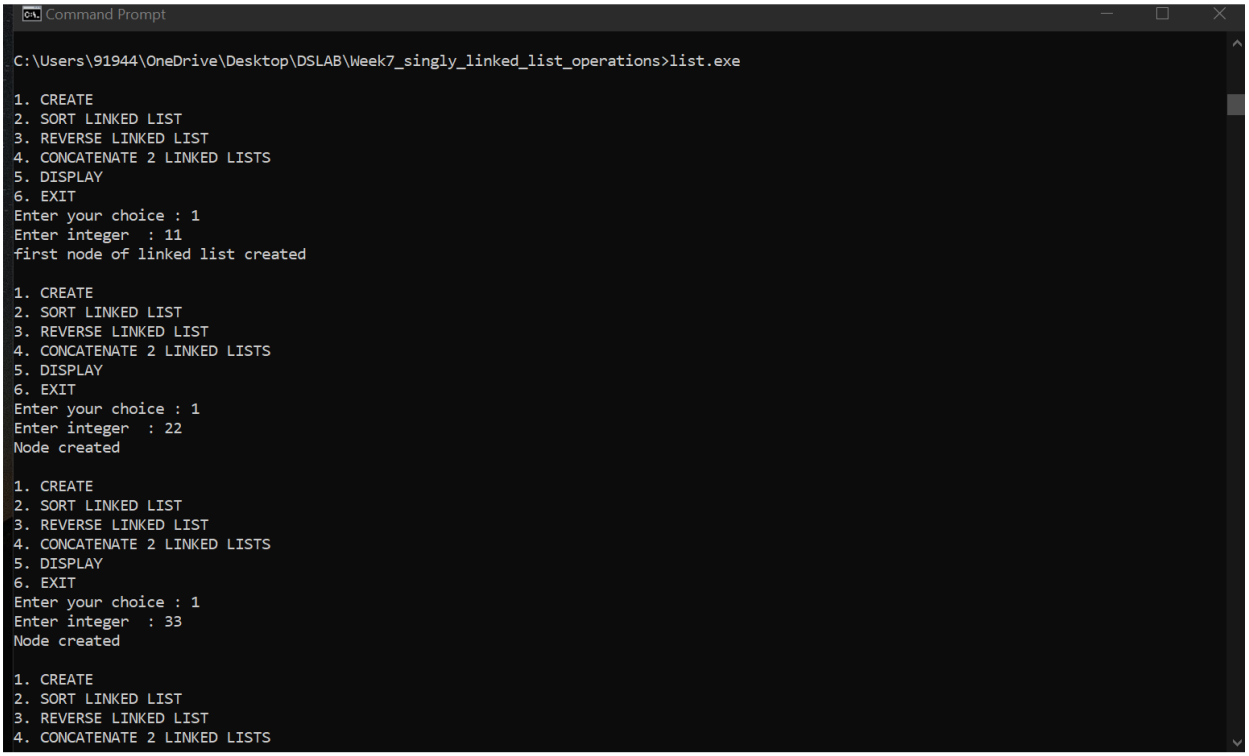
```
#include <stdio.h>
#include <stdlib.h>
void create1();
void create2();
void sort();
void reverse();
void concatenate();
void display();
struct node
{
  int data;
  struct node *next;
};

struct node *head=NULL;
struct node *head2= NULL;
int c;

int main(int argc, char **argv)
{
        int choice;
   do
   {
   printf("\n1. CREATE \n2. SORT LINKED LIST \n3. REVERSE LINKED LIST \n4. CONCATENATE 2
LINKED LISTS \n5. DISPLAY  \n6. EXIT");
   printf("\nEnter your choice : ");
   scanf("%d",&choice);
   switch(choice)
   {
      case 1: create1();
            break;
      case 2: sort();
            break;
      case 3: reverse();
            break;
      case 4: create2();
            concatenate();
            break;
      case 5: display();
            break;
      case 6: printf("\nEXITINGGG....\n");
            break;
   }
   }while(choice != 6);
   return 0;
}

void create1()
{
```

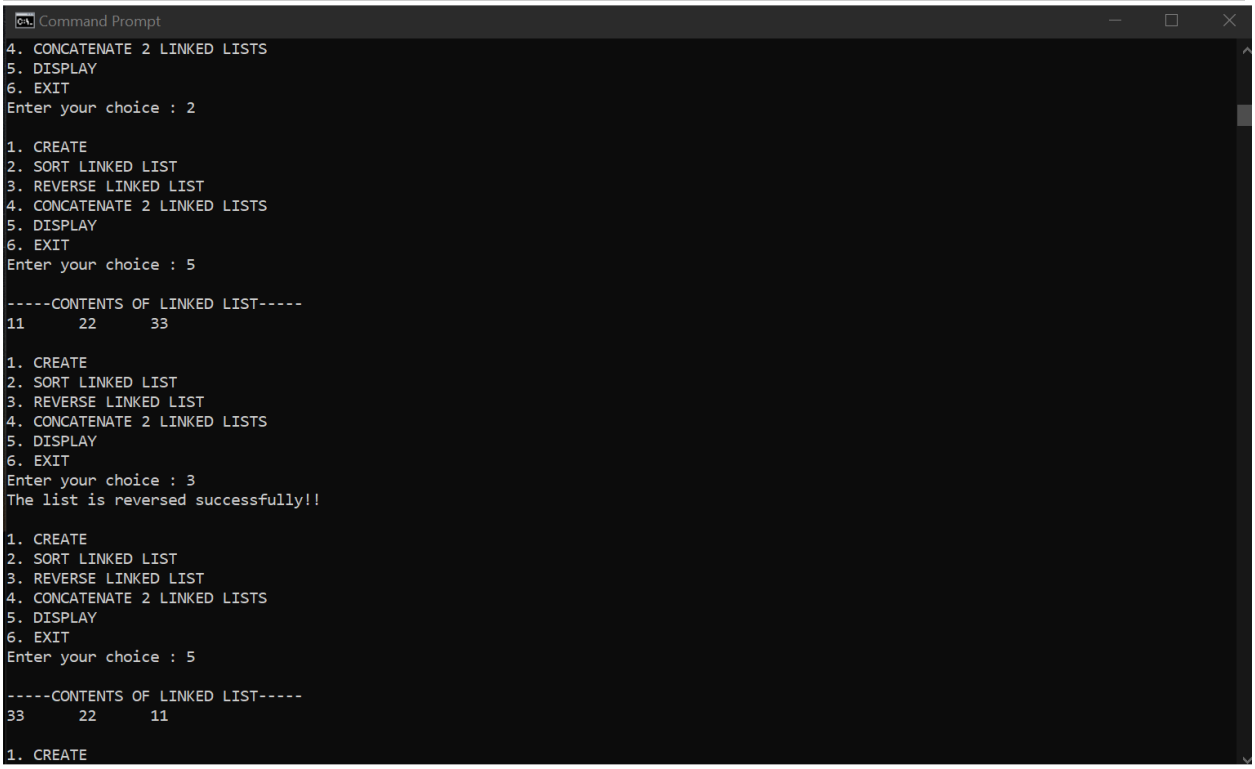
```
   struct node *newnode;
   struct node *temp;
   int s;
   printf("Enter integer  : ");
   scanf("%d",&s);
   newnode=(struct node*)malloc(sizeof(struct node));
   newnode->data =s;
   if (head==NULL)
   {
     newnode->next=NULL;
     head=newnode;
     printf("first node of linked list created\n");
     c++;
   }
    else
    {
       temp=head;
       while(temp->next!=NULL)
       {
          temp=temp->next;
       }
   temp->next=newnode;
   newnode->next=NULL;
   c++;
   printf("Node created\n");
  }
}

void reverse()
{
   struct node *prev=NULL,*current=head, *next=NULL;
   while(current!=NULL)
   {
      next=current->next;
      current->next=prev;
      prev=current;
      current=next;
   }
   head=prev;
   printf("The list is reversed successfully!!\n");
}

void display()
{
   struct node *ptr=NULL;
   ptr=head;

   if(ptr==NULL)
   {
      printf("Nothing to print\n");
   }
   else
```

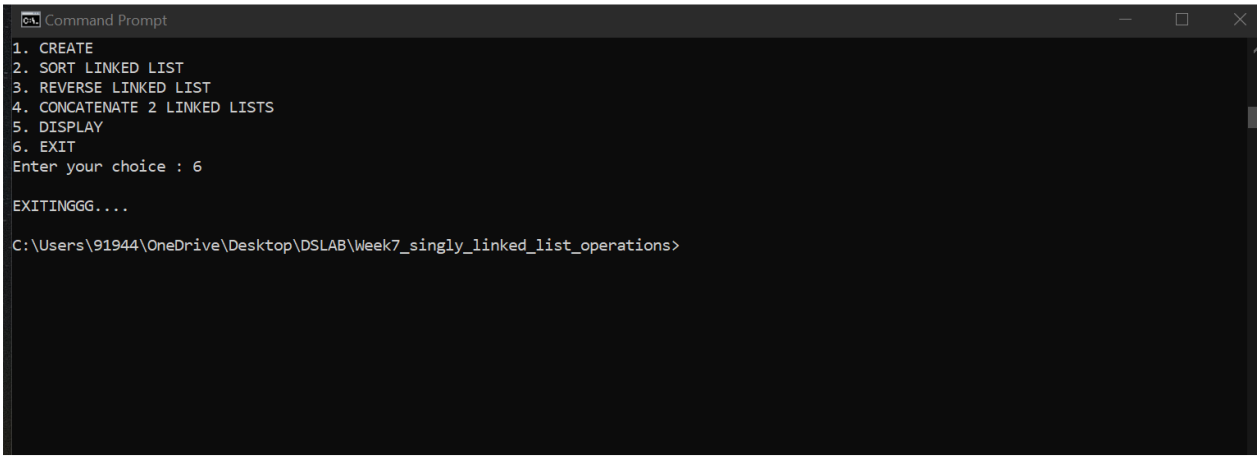
```
    {
       printf("\n-----CONTENTS OF LINKED LIST-----\n");
       while(ptr!=NULL)
      {
        printf("%d\t",ptr->data);
        ptr=ptr->next;
      }
    }
    printf("\n");
}

void concatenate()
{
    struct node *ptr;
       if(head==NULL)
       {
             head=head2;
       }
       if(head2==NULL)
            {
             head2=head;
            }
       ptr=head;
       while(ptr->next!=NULL)
             ptr=ptr->next;
       ptr->next=head2;
}

void sort()
{
    int swap, i;
    struct node *ptr1;
    struct node *lptr = NULL;


    if (head == NULL)
       return;

    do
    {
       swap = 0;
       ptr1 = head;

       while (ptr1->next != lptr)
       {
          if (ptr1->data > ptr1->next->data)
          {
             int temp = ptr1->data;
             ptr1->data = ptr1->next->data;
             ptr1->next->data = temp;
             swap = 1;
          }
```

```
            ptr1 = ptr1->next;
        }
        lptr = ptr1;
    }
    while(swap);
}

void create2()
{
  struct node *newnode;
  struct node *temp;
    int s,y;
    printf("Enter elements to the SECOND linked list 2\n");
    do
    {
    printf("Enter integer  : \n");
    scanf("%d",&s);
    newnode=(struct node*)malloc(sizeof(struct node));
    newnode->data =s;
    if (head2==NULL)
    {
      newnode->next=NULL;
      head2=newnode;
      printf("first node of linked list created\n");
      c++;
    }
     else
     {
    temp=head2;
        while(temp->next!=NULL)
        {
      temp=temp->next;
        }
    temp->next=newnode;
    newnode->next=NULL;
    c++;
    printf("Node created\n");
   }
   printf("Do u want to continue adding: 0 or 1\n");
   scanf("%d",&y);
    }
    while(y!=0);
}
```

OUTPUT:


Command Prompt
C:\Users\91944\OneDrive\Desktop\DSLAB\Week7_singly_linked_list_operations>list.exe
1. CREATE
2. SORT LINKED LIST
3. REVERSE LINKED LIST
4. CONCATENATE 2 LINKED LISTS
5. DISPLAY
6. EXIT
Enter your choice : 1
Enter integer : 11
first node of linked list created
1. CREATE
2. SORT LINKED LIST
3. REVERSE LINKED LIST
4. CONCATENATE 2 LINKED LISTS
5. DISPLAY
6. EXIT
Enter your choice : 1
Enter integer : 22
Node created
1. CREATE
2. SORT LINKED LIST
3. REVERSE LINKED LIST
4. CONCATENATE 2 LINKED LISTS
5. DISPLAY
6. EXIT
Enter your choice : 1
Enter integer : 33
Node created
1. CREATE
2. SORT LINKED LIST
3. REVERSE LINKED LIST
4. CONCATENATE 2 LINKED LISTS


Command Prompt
4. CONCATENATE 2 LINKED LISTS
5. DISPLAY
6. EXIT
Enter your choice : 2
1. CREATE
2. SORT LINKED LIST
3. REVERSE LINKED LIST
4. CONCATENATE 2 LINKED LISTS
5. DISPLAY
6. EXIT
Enter your choice : 5
-----CONTENTS OF LINKED LIST-----
11 22 33
1. CREATE
2. SORT LINKED LIST
3. REVERSE LINKED LIST
4. CONCATENATE 2 LINKED LISTS
5. DISPLAY
6. EXIT
Enter your choice : 3
The list is reversed successfully!!
1. CREATE
2. SORT LINKED LIST
3. REVERSE LINKED LIST
4. CONCATENATE 2 LINKED LISTS
5. DISPLAY
6. EXIT
Enter your choice : 5
-----CONTENTS OF LINKED LIST-----
33 22 11
1. CREATE

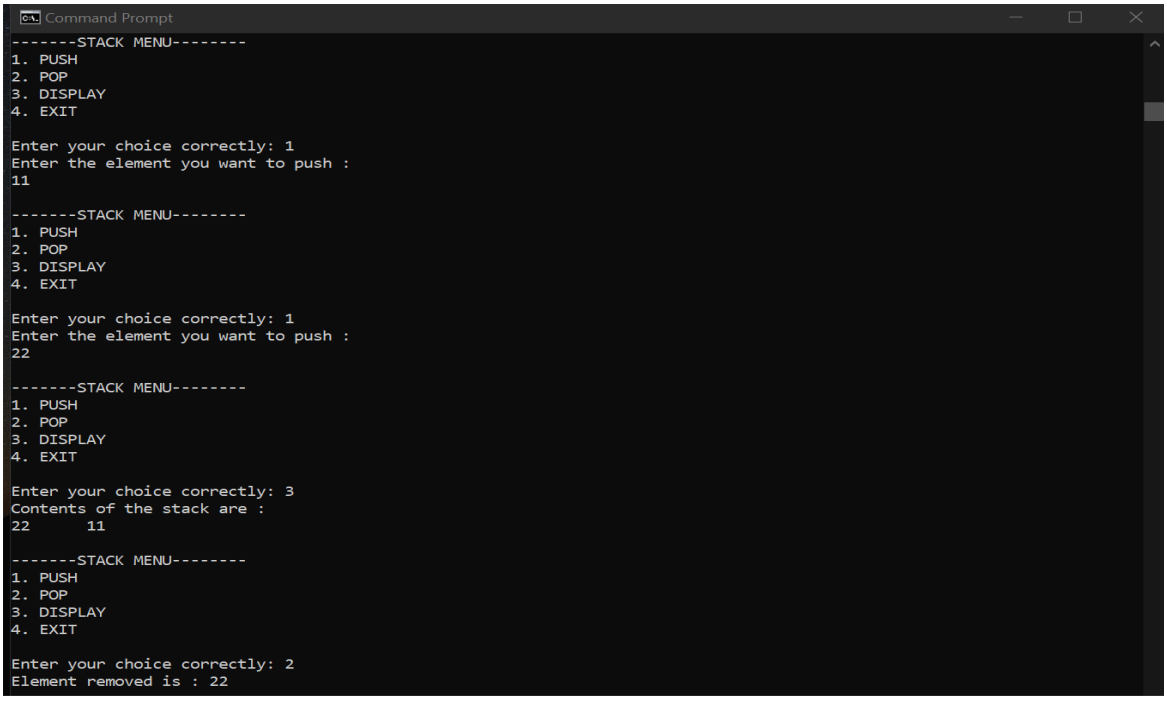

Command Prompt
1. CREATE
2. SORT LINKED LIST
3. REVERSE LINKED LIST
4. CONCATENATE 2 LINKED LISTS
5. DISPLAY
6. EXIT
Enter your choice : 6
EXITINGGG....
C:\Users\91944\OneDrive\Desktop\DSLAB\Week7_singly_linked_list_operations>

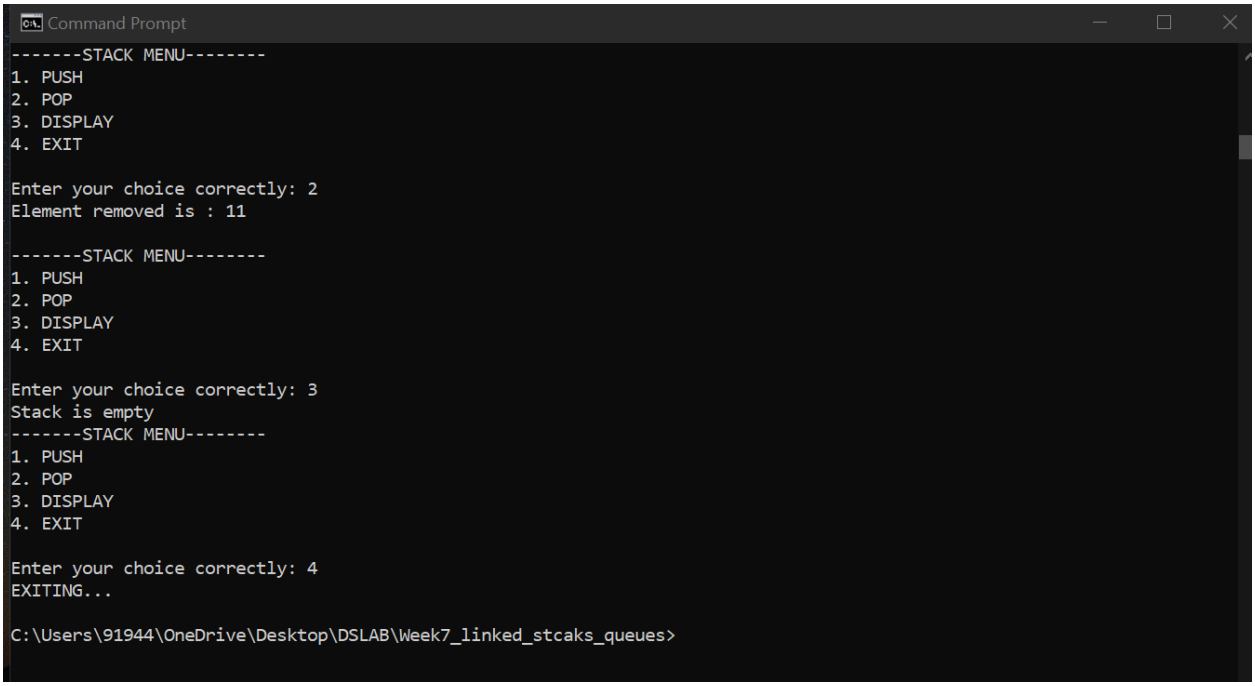
LAB PROGRAM – 8
INPUT FOR STACK USING SINGLY LINKED LIST:

```
#include <stdio.h>
#include <stdlib.h>

void push();
void pop();
void display();
struct node
{
   int data;
   struct node *next;
};
struct node *top=NULL;

int main()
{
        int choice;
   do
   {
      printf("\n-------STACK MENU--------");
      printf("\n1. PUSH \n2. POP \n3. DISPLAY \n4. EXIT \n");
      printf("\nEnter your choice correctly: ");
      scanf("%d",&choice);
      switch(choice)
      {
         case 1: push(); break;
         case 2: pop(); break;
         case 3: display();break;
         case 4: printf("EXITING...\n"); break;
         default: printf("INVALID CHOICE!!!\n");
      }
   }while(choice != 4);
   return 0;
}


void push()
{
   int item;
   struct node *newnode;
   printf("Enter the element you want to push : \n");
   scanf("%d",&item);
   newnode=(struct node*)malloc(sizeof(struct node));
   newnode->data=item;
   newnode->next=NULL;
   if(top==NULL)
      top=newnode;
   else
   {
      newnode->next=top;
```

```
        top=newnode;
    }
}

void pop()
{
    if(top==NULL)
        printf("Stack is empty!!\n");
    else
    {
      printf("Element removed is : %d\n", top->data);
      top=top->next;
    }
}

void display()
{
    struct node *temp;
    temp=top;
    if(top==NULL)
    {
        printf("Stack is empty");
        return;
    }
    printf("Contents of the stack are : \n");
    while(temp!=NULL)
    {
        printf("%d\t",temp->data);
        temp=temp->next;
    }
    printf("\n");
}
```

OUTPUT FOR STACK USING SINGLY LINKED LIST:





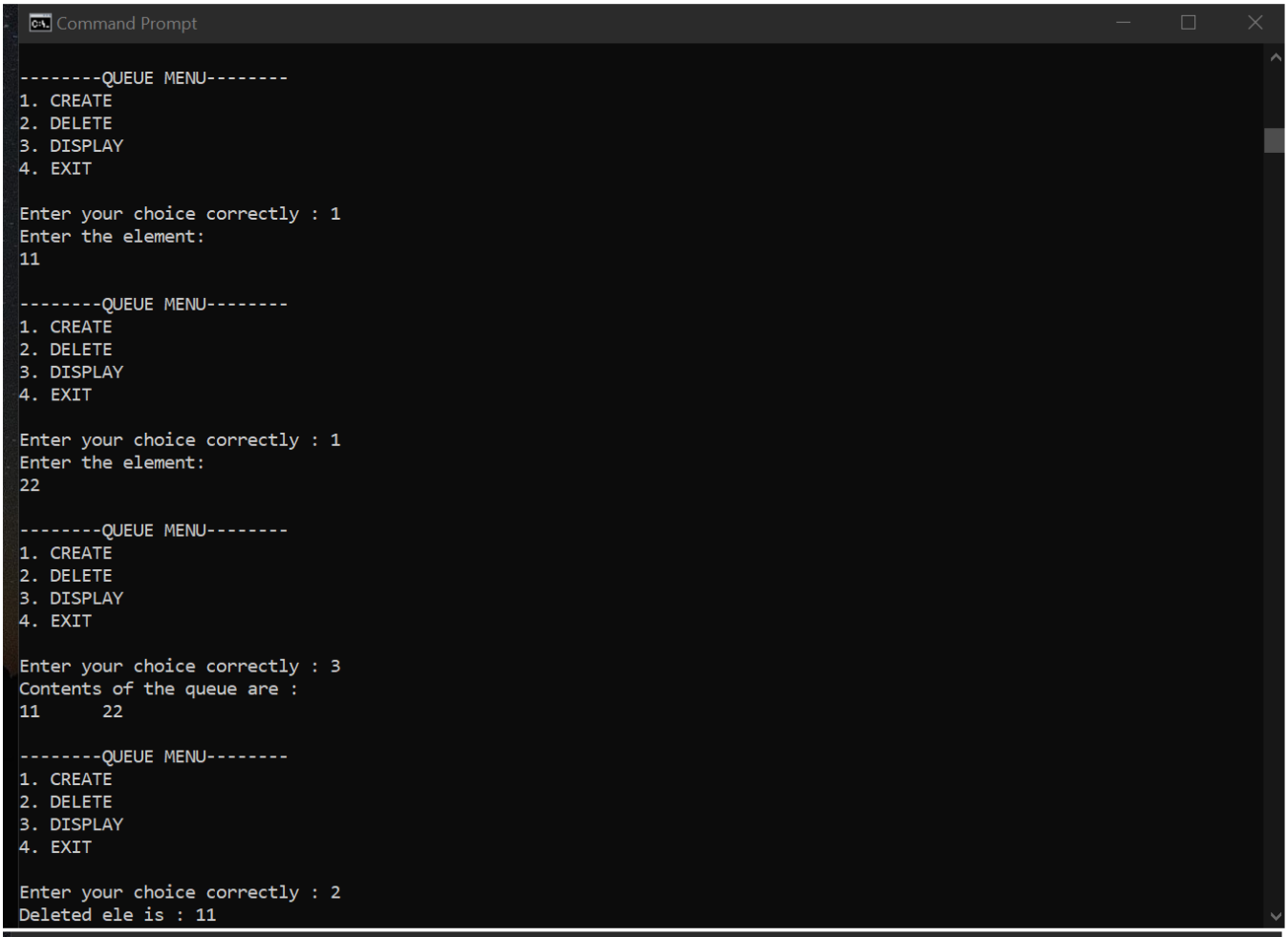
INPUT FOR QUEUE USING SINGLY LINKED LIST:

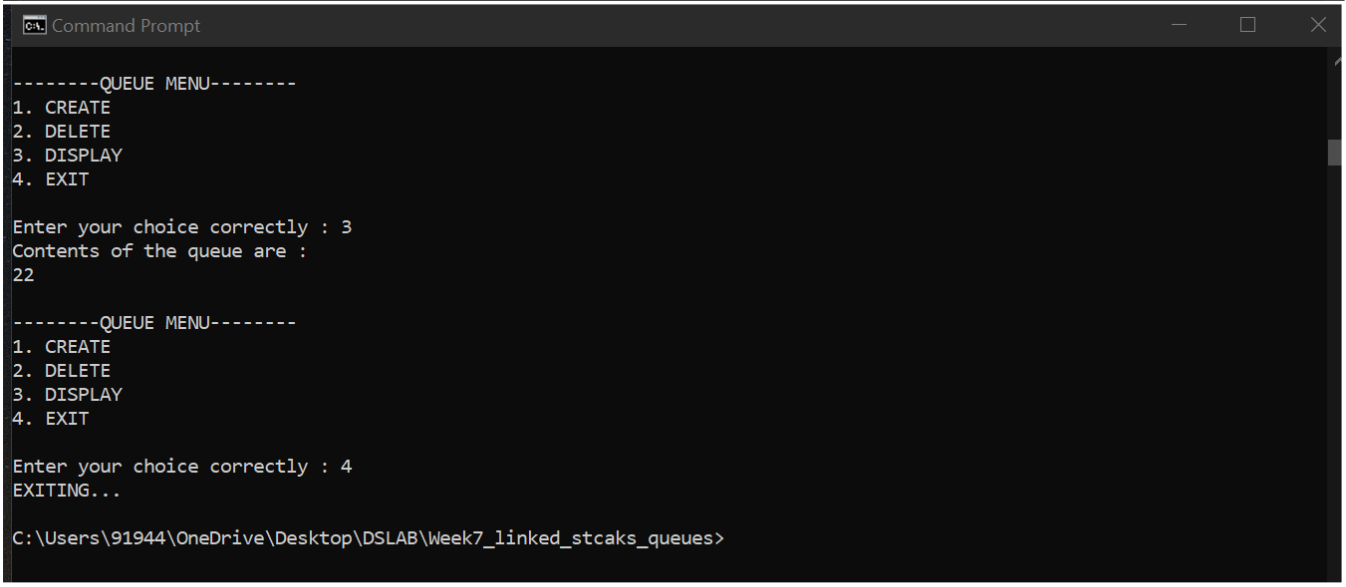
```
#include <stdio.h>
#include <stdlib.h>
struct node
{
   int data;
   struct node *next;
};

void insert();
void display();
void delete();

struct  node *rear=NULL, *front =NULL;

int main()
{
;
   int choice;
   do
   {
      printf("\n--------QUEUE MENU--------");
      printf("\n1. CREATE \n2. DELETE  \n3. DISPLAY  \n4. EXIT \n");
      printf("\nEnter your choice correctly : ");
      scanf("%d",&choice);
      switch(choice)
      {
         case 1: insert(); break;
         case 2: delete(); break;
         case 3: display();break;
         case 4: printf("EXITING...\n"); break;
         default: printf("INVALID CHOICE!!!\n");
```

```
      }
   }while(choice != 4);
   return 0;
}

void insert()
{
   struct  node *newnode;
   newnode=(struct node *) malloc(sizeof(struct node));
   printf("Enter the element:\n");
   scanf("%d",&newnode->data);
   newnode->next=NULL;

   if(rear==NULL)
   {
      rear=newnode;
      front=newnode;
   }
   else
   {
      rear->next=newnode;
      rear=newnode;
   }
}

void delete()
{
   if(front==NULL)
   {
     printf("Queue is empty!! QUEUE UNDERFLOW!!\n");return;
   }

   else
   {
      printf("Deleted ele is : %d\n",front->data);
      if(front==rear)
      {
        front=NULL;
        rear=NULL;
      }
      else
      front=front->next;
   }
}

void display()
{
   struct node *temp;
   if(front ==NULL)
   {
      printf("Queue is empty!! NOTHING TO DISPLAY!!\n");
      return;
```

```
    }
    temp=front;
    printf("Contents of the queue are : \n");
    while (temp !=NULL)
    {
        printf("%d\t",temp->data);
        temp=temp->next;
    }
    printf("\n");
}
```

OUTPUT FOR QUEUE USING SINGLY LINKED LIST:





LAB PROGRAM – 9

INPUT:

```
#include <stdio.h>
#include <stdlib.h>

struct node
{
        int data;
        struct node *next;
        struct node *prev;
};

struct node *head=NULL;

void insert_beg()
{
         struct node *new_node;
         new_node=(struct node*)malloc(sizeof(struct node));
         printf("Enter the number you want to insert : \n");
         scanf("%d",&new_node->data);
         new_node->next=NULL;
         new_node->prev=NULL;

         if(head==NULL)
         {
                        head=new_node;
    printf("Element INSERTED!!\n");
         }
         else
         {
                        new_node->next=head;
                        head->prev=new_node;
                        head=new_node;
    printf("Element INSERTED!!\n");
         }
}

void insert_left()
{
        int listele;
        struct node *new_node,*temp;
        printf("Enter the element in the list : \n");
        scanf("%d",&listele);
        new_node=(struct node*)malloc(sizeof(struct node));
        printf("Enter the new node data : \n");
        scanf("%d",&new_node->data);
        new_node->next=NULL;
        new_node->prev=NULL;
  if(head==NULL)
  {
     printf("Empty list\n"); return;
```

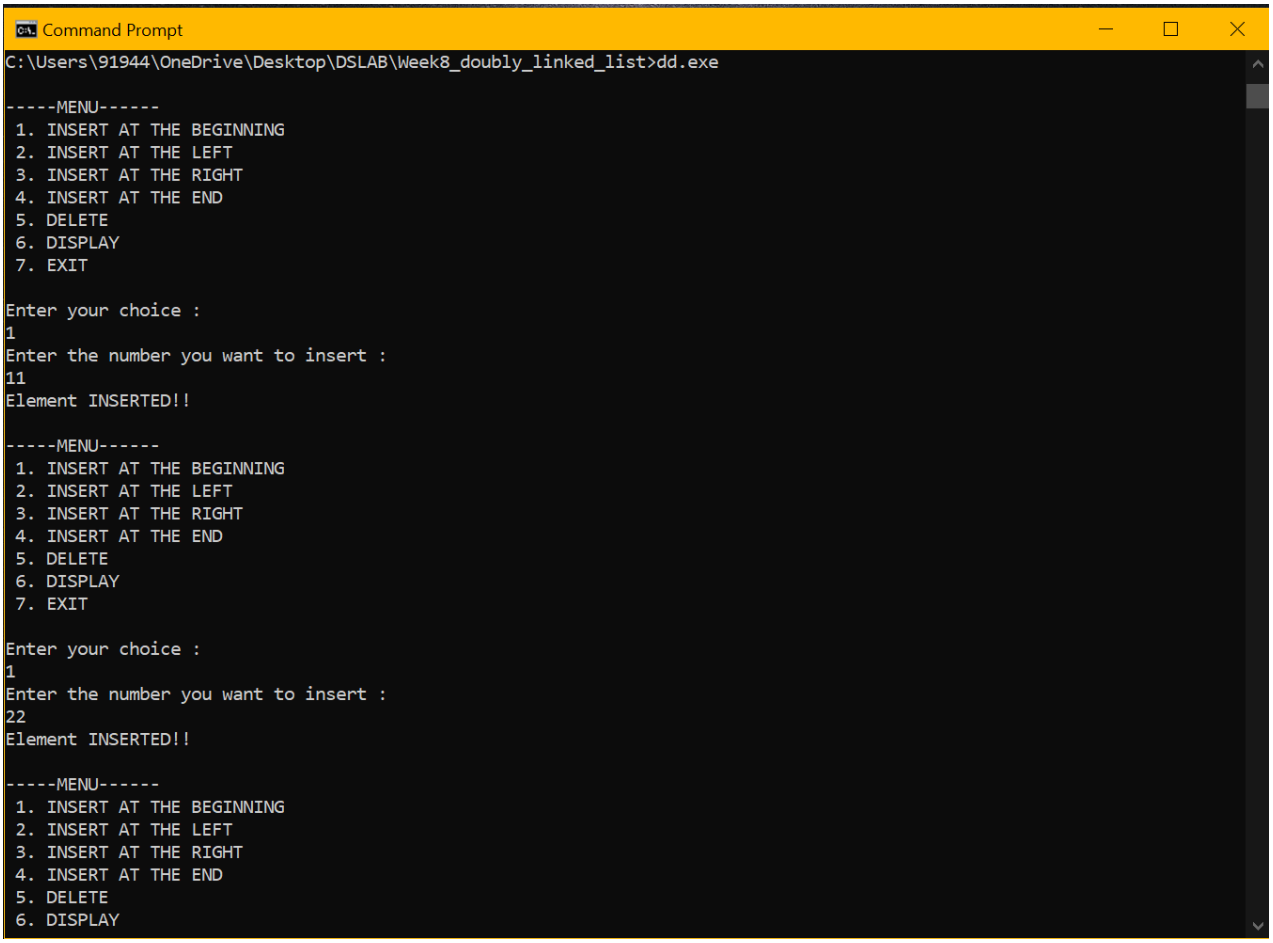
```
   }
        temp=head;
        while(temp->data!=listele)
        {
                temp=temp->next;
                if(temp==NULL)
                {
                        printf("Element is  not in the list");
                        return;
                }
        }
  new_node->prev=temp->prev;
        temp->prev=new_node;
        new_node->next=temp;
        new_node->prev->next=new_node;
}

void insert_right()
{
  int listele;
        struct node *new_node,*temp;
        printf("Enter the element in the list :\n");
        scanf("%d",&listele);
        new_node=(struct node*)malloc(sizeof(struct node));
        printf("Enter the new node data : \n");
        scanf("%d",&new_node->data);
        new_node->next=NULL;
        new_node->prev=NULL;
    if(head==NULL)
    {
       printf("Empty list\n"); return;
    }
        temp=head;
        while(temp->data!=listele)
        {
                temp=temp->next;
                if(temp==NULL)
                {
                        printf("Element is  not in the list");
                        return;
                }
        }
        new_node->next=temp->next;
        temp->next=new_node;
        new_node->prev=temp;
        new_node->next->prev=new_node;
}

void insert_end()
{
        struct node *new_node,*temp;
        new_node=(struct node*)malloc(sizeof(struct node));
```

```
        printf("Enter the number you want to insert : \n");
        scanf("%d",&new_node->data);
        new_node->next=NULL;
        new_node->prev=NULL;
        if(head==NULL)
        {
                head=new_node;
        }
        else
        {
                 temp=head;
                 while(temp->next!=NULL)
                 temp=temp->next;
                 temp->next=new_node;
                 new_node->prev=temp;
        }
}

void del()
{
        struct node *temp;
        int ele;
  if(head==NULL)
  {
     printf("EMPTY LIST!!\n");
     return;
  }
        printf("Enter the element to be deleted :\n");
        scanf("%d",&ele);
        temp=head;
        while(temp->data!=ele)
        {
                temp=temp->next;
                if(temp==NULL)
                {
                 printf("Element is NOT FOUND in the list!!\n");
                 return;
                }
         }
         if(temp==head)
         {
                 head=head->next;
    printf("Element is DELETED!!\n");
         }
         else if(temp->next==NULL)
         {
                        temp=temp->prev;
                        temp->next=NULL;
     printf("Element is DELETED!!\n");
         }
         else
         {
```

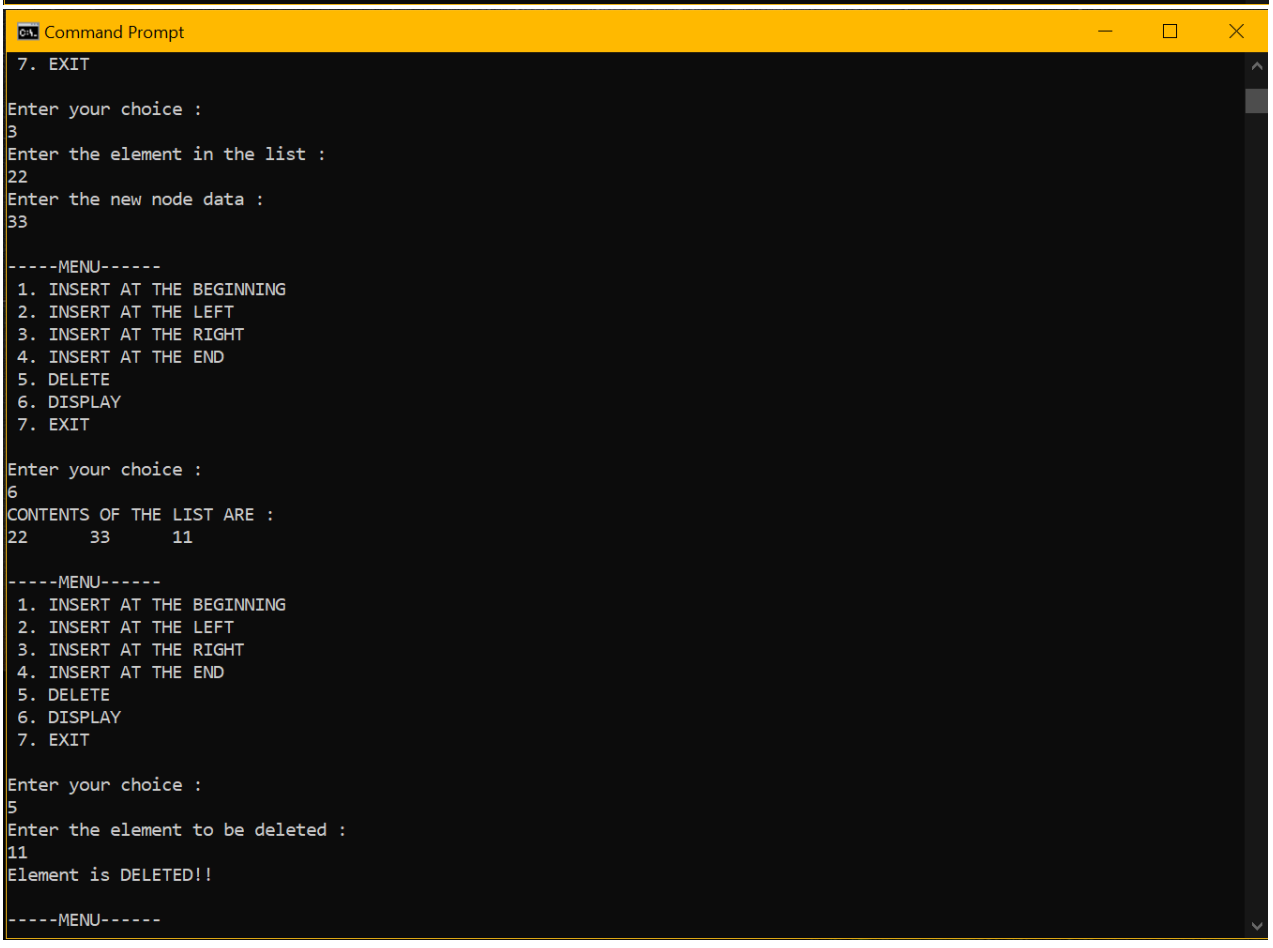
```
                temp->prev->next=temp->next;
                temp->next->prev=temp->prev;
    printf("Element is DELETED!!\n");
        }
}

void display()
{
        struct node *temp;
        temp=head;
  if(temp==NULL)
  {
     printf("Nothing to print!! EMPTY LIST!!\n");
     return;
  }
  printf("CONTENTS OF THE LIST ARE : \n");
        while(temp!=NULL)
        {
                printf("%d\t",temp->data);
                temp=temp->next;
        }
        printf("\n");
}

int main()
{
        int choice;
        do
        {
    printf("\n-----MENU------\n");
    printf(" 1. INSERT AT THE BEGINNING \n");
                        printf(" 2. INSERT AT THE LEFT \n");
    printf(" 3. INSERT AT THE RIGHT \n");
    printf(" 4. INSERT AT THE END \n");
                        printf(" 5. DELETE \n");
                        printf(" 6. DISPLAY\n");
                        printf(" 7. EXIT\n");
                        printf("\nEnter your choice : \n");
                        scanf("%d",&choice);
                        switch(choice)
                        {
      case 1 : insert_beg(); break;
                                case 2 : insert_left(); break;
      case 3 : insert_right(); break;
      case 4 : insert_end(); break;
                                case 5 : del(); break;
                                case 6 : display(); break;
                                case 7 : printf("EXITING...\n");break;
      default : printf("Enter your choice CORRECTLY!!\n"); break;
                        }
        }
  while(choice != 7);
```

```
    return 0;
}
```

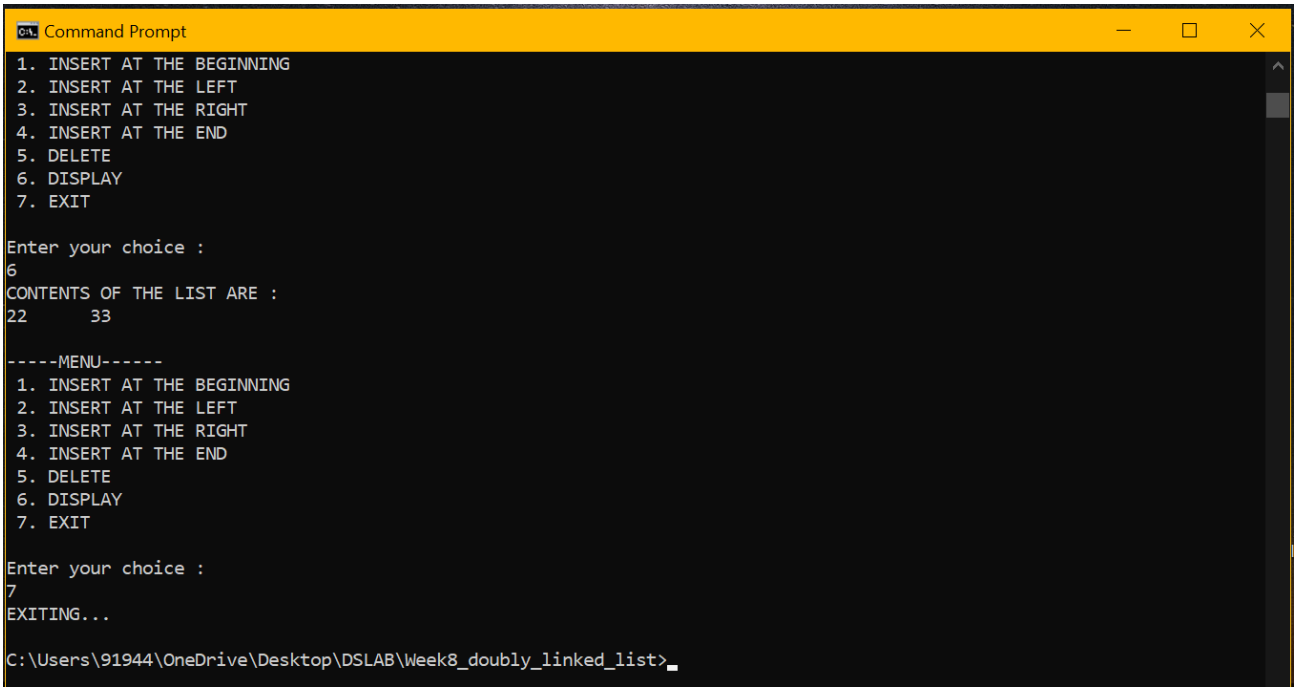
OUTPUT:



```
Command Prompt
C:\Users\91944\OneDrive\Desktop\DSLAB\Week8_doubly_linked_list>dd.exe

-----MENU------
 1. INSERT AT THE BEGINNING
 2. INSERT AT THE LEFT
 3. INSERT AT THE RIGHT
 4. INSERT AT THE END
 5. DELETE
 6. DISPLAY
 7. EXIT

Enter your choice :
1
Enter the number you want to insert :
11
Element INSERTED!!

-----MENU------
 1. INSERT AT THE BEGINNING
 2. INSERT AT THE LEFT
 3. INSERT AT THE RIGHT
 4. INSERT AT THE END
 5. DELETE
 6. DISPLAY
 7. EXIT

Enter your choice :
1
Enter the number you want to insert :
22
Element INSERTED!!

-----MENU------
 1. INSERT AT THE BEGINNING
 2. INSERT AT THE LEFT
 3. INSERT AT THE RIGHT
 4. INSERT AT THE END
 5. DELETE
 6. DISPLAY
```



```
Command Prompt
 7. EXIT

Enter your choice :
3
Enter the element in the list :
22
Enter the new node data :
33

-----MENU------
 1. INSERT AT THE BEGINNING
 2. INSERT AT THE LEFT
 3. INSERT AT THE RIGHT
 4. INSERT AT THE END
 5. DELETE
 6. DISPLAY
 7. EXIT

Enter your choice :
6
CONTENTS OF THE LIST ARE :
22      33      11

-----MENU------
 1. INSERT AT THE BEGINNING
 2. INSERT AT THE LEFT
 3. INSERT AT THE RIGHT
 4. INSERT AT THE END
 5. DELETE
 6. DISPLAY
 7. EXIT

Enter your choice :
5
Enter the element to be deleted :
11
Element is DELETED!!

-----MENU------
```



LAB PROGRAM – 10

INPUT:

```
#include<stdio.h>
#include<stdlib.h>
struct node
{
    int data;
    struct node* left;
    struct node* right;
}*root1;

struct node *create()
{
        struct node *temp;
        printf("Enter ROOT NODE ELEMENT : ");
        temp=(struct node*)malloc(sizeof(struct node));
        scanf("%d",&temp->data);
        temp->left=temp->right=NULL;
        return temp;
}

void insert(struct node *root,struct node *temp)
{
        if(temp->data<root->data)
        {
                if(root->left!=NULL)
                        insert(root->left,temp);
                else
                        root->left=temp;
        }
        if(temp->data>root->data)
```

```
        {
                if(root->right!=NULL)
                        insert(root->right,temp);
                else
                        root->right=temp;
        }
}


void printPostorder(struct node* node)
{
    if (node == NULL)
      return;
    printPostorder(node->left);
    printPostorder(node->right);
    printf("%d\t", node->data);
}


void printInorder(struct node* node)
{
    if (node == NULL)
        return;
    printInorder(node->left);
    printf("%d\t", node->data);
    printInorder(node->right);
}


void printPreorder(struct node* node)
{
    if (node == NULL)
        return;
    printf("%d\t", node->data);
    printPreorder(node->left);
    printPreorder(node->right);
}

int main()
{
        int choice;
        struct node* temp;
        do
        {
                printf("\n-----MENU-----\n");
                printf("1. CREATE\n");
                printf("2. INSERT\n");
                printf("3. PREORDER TRAVERSAL\n");
                printf("4. INORDER TRAVERSAL\n");
                printf("5. POSTORDER TRAVERSAL\n");
                printf("6. EXIT\n");
                printf("Enter your choice correctly : \n");
```

```
            scanf("%d", &choice);
            switch(choice)
            {
                case 1: root1 = create();
                        break;
                case 2: printf("Enter the VALUE you want to INSERT : ");
                        temp=(struct node*)malloc(sizeof(struct node));
                        scanf("%d",&temp->data);
                        insert(root1, temp);
                        break;
                case 3: printPreorder(root1);
                        break;
                case 4: printInorder(root1);
                        break;
                case 5: printPostorder(root1);
                        break;
                case 6: printf("EXITING...!!!");
                        break;
                default: printf("Incorrect Choice!!\n");
            }
        }while(choice != 6);
        return 0;
}
```

OUTPUT:

```
-----MENU-----
1. CREATE
2. INSERT
3. PREORDER TRAVERSAL
4. INORDER TRAVERSAL
5. POSTORDER TRAVERSAL
6. EXIT
Enter your choice correctly :
1
Enter ROOT NODE ELEMENT : 9

-----MENU-----
1. CREATE
2. INSERT
3. PREORDER TRAVERSAL
4. INORDER TRAVERSAL
5. POSTORDER TRAVERSAL
6. EXIT
Enter your choice correctly :
2
Enter the VALUE you want to INSERT : 4
```

```
-----MENU-----
1. CREATE
2. INSERT
3. PREORDER TRAVERSAL
4. INORDER TRAVERSAL
5. POSTORDER TRAVERSAL
6. EXIT
Enter your choice correctly :
2
Enter the VALUE you want to INSERT : 15

-----MENU-----
1. CREATE
2. INSERT
3. PREORDER TRAVERSAL
4. INORDER TRAVERSAL
5. POSTORDER TRAVERSAL
6. EXIT
Enter your choice correctly :
2
Enter the VALUE you want to INSERT : 6

-----MENU-----
1. CREATE
2. INSERT
3. PREORDER TRAVERSAL
4. INORDER TRAVERSAL
5. POSTORDER TRAVERSAL
6. EXIT
Enter your choice correctly :
2
Enter the VALUE you want to INSERT : 12

-----MENU-----
1. CREATE
2. INSERT
3. PREORDER TRAVERSAL
4. INORDER TRAVERSAL
5. POSTORDER TRAVERSAL
6. EXIT
Enter your choice correctly :
2
Enter the VALUE you want to INSERT : 17

-----MENU-----
1. CREATE
2. INSERT
3. PREORDER TRAVERSAL
4. INORDER TRAVERSAL
5. POSTORDER TRAVERSAL
6. EXIT
Enter your choice correctly :
```

```
2
Enter the VALUE you want to INSERT : 2

-----MENU-----
1. CREATE
2. INSERT
3. PREORDER TRAVERSAL
4. INORDER TRAVERSAL
5. POSTORDER TRAVERSAL
6. EXIT
Enter your choice correctly :
3
9       4       2       6       15      12      17
-----MENU-----
1. CREATE
2. INSERT
3. PREORDER TRAVERSAL
4. INORDER TRAVERSAL
5. POSTORDER TRAVERSAL
6. EXIT
Enter your choice correctly :
4
2       4       6       9       12      15      17
-----MENU-----
1. CREATE
2. INSERT
3. PREORDER TRAVERSAL
4. INORDER TRAVERSAL
5. POSTORDER TRAVERSAL
6. EXIT
Enter your choice correctly :
5
2       6       4       12      17      15      9
-----MENU-----
1. CREATE
2. INSERT
3. PREORDER TRAVERSAL
4. INORDER TRAVERSAL
5. POSTORDER TRAVERSAL
6. EXIT
Enter your choice correctly :
6
EXITING...!!!

...Program finished with exit code 0
Press ENTER to exit console.
```