**PROGRAMS: Implementation of Stack Linked Lists**

**Input:**

#include<stdio.h>

#include<stdlib.h>

struct node

{

   int data;

   struct node\*link;

};

struct node \*top=NULL;

 void push()

 {

   int x;

   printf("Enter the element to be inserted:");

   scanf("%d",&x);

   struct node\*temp;

   temp=(struct node\*)malloc(sizeof(struct node));

   if(temp==NULL)

   {

    printf("Stack Overflow");

   }

   else

   {

    temp->data=x;

    temp->link=top;

    top=temp;

    printf("Element %d inserted \n",x);

   }

 }

 void pop()

 {

    struct node\*p;

    if(top==NULL)

    {printf("Stack Underflow\n");}

    else

    {

      printf("Element %d deleted\n", top->data);

Expt no: 5

      p=top;

      top=top->link;

      free(p);

    }

 }

 void display()

 {

   struct node\*p;

   if(top==NULL)

   {

     printf("List is Empty\n");

     return;

   }

   p=top;

   printf("Stack:\t");

   while(p!=NULL)

   {

    printf("%d\t",p->data);

    p=p->link;

   }

   printf("\n");

 }

 void peek()

 {

   if(top==NULL)

   {

     printf("Stack Underflow\n");

   }

  else

  {

   printf("The top element is %d\n",top->data);

  }

 }

void underflow()

{

  if(top==NULL)

  {

    printf("Stack Underflow\n");

    return;

  }

  else

  {

   printf("Stack not Underflow\n");

  }

}

void count()

{

    struct node \*p;

    int count=0;

    p=top;

    while(p!=NULL)

    {

        p=p->link;

        count++;

    }

    printf("count = %d\n",count);

}

int main()

{

  int choice;

  printf("1.Push\n2.Pop\n3.Display\n4.Peek\n5.Underflow\n6.Count\n7.Exit");

  while(1){

      printf("\nEnter choice:");

      scanf("%d",&choice);

       switch(choice)

       {

          case 1:push();

                  break;

          case 2:pop();

                  break;

          case 3:display();

                  break;

          case 4:peek();

                  break;

          case 5:underflow();

                  break;

          case 6:count();

Expt no: 5

                break;

          case 7:exit(0);

          default:printf("invalid choice\n");

       }

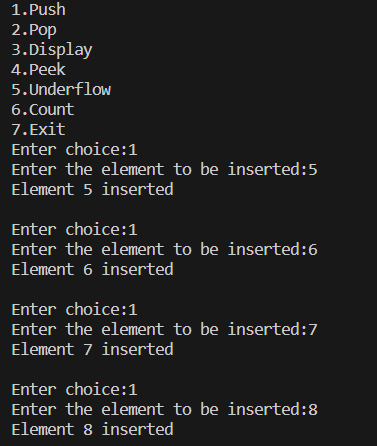
  }

    return 0;

}

**Output:**

1. To push elements on the stacks:



1. To display the stack:



1. To peek the top element:



1. To view the count:

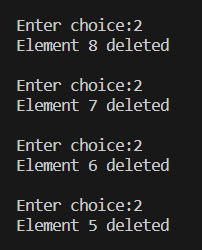


1. To check for stack underflow:



1. To pop elements from the stack:

Expt no: 5



1. To check for stack underflow:



1. To pop an element from an empty stack:



1. To display the stack (empty stack):



1. To peek the top element (empty stack):

****

1. to view the count (empty stack):



**PROGRAMS: Implementation of Queue Linked Lists**

**Input:**

#include<stdio.h>

#include<stdlib.h>

struct node

{

    int data;

    struct node \*link;

};

struct node \*front=NULL;

struct node \*rear=NULL;

void insert()

{

    int x;

    printf("Enter the element to be inserted:");

    scanf("%d",&x);

    struct node \*tmp;

    tmp=(struct node\*)malloc(sizeof(struct node));

    if(tmp==NULL)

    {

        printf("No space for Dynamic Memory Allocation\n");

    }

    else

    {

        tmp->data=x;

        tmp->link=NULL;

        if(front==NULL)

        {

            front=tmp;

        }

        else

        {

            rear->link=tmp;

        }

        rear=tmp;

        printf("Element inserted successfully\n");

    }

Expt no: 5

}

void delete()

{

    struct node \*p;

    if(front==NULL)

    {

        printf("Queue Underflow\n");

    }

    else

    {

        printf("%d deleted successfully\n",front->data);

        p=front;

        front=front->link;

        free(p);

    }

}

void display()

{

    struct node \*tmp;

    if(front==NULL)

    {

        printf("Queue is empty\n");

    }

    tmp=front;

    {

        if(front!=NULL)

            printf("Queue:\t");

        while(tmp!=NULL)

        {

            printf("%d\t",tmp->data);

            tmp=tmp->link;

        }

    }

    printf("\n");

}

void peek()

{

    if(front==NULL)

    {

        printf("Queue underflow\n");

    }

    else

    {

        printf("Front element is %d\n",front->data);

    }

}

void underflow()

{

    if(front==NULL)

    printf("Queue underflow\n");

    else

    printf("No Queue Underflow\n");

}

int main()

{

    int c, i;

    printf("\n1.Insert\n2.Delete\n3.Display\n4.Peek\n5.Check Queue Underflow Condition\n6.Exit\n");

    while(1)

    {

        printf("\nEnter choice:");

        scanf("%d", &c);

        switch(c)

        {

            case 1:insert();

                   break;

            case 2:delete();

                   break;

            case 3:display();

                   break;

            case 4:peek();

                   break;

            case 5:underflow();

                   break;

            case 6:exit(0);

                   break;

            default:printf("Invalid choice\n");

                    break;

Expt no: 5

            }

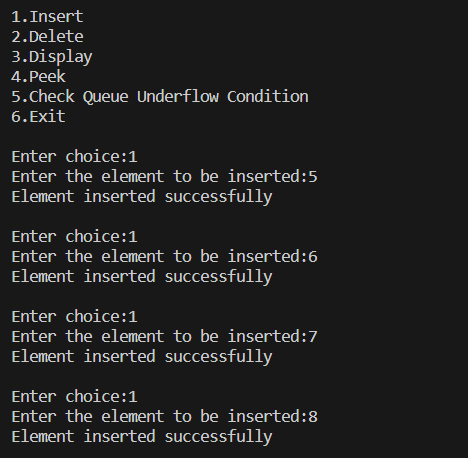
    }

return 0;

}

**Output:**

1. To insert elements in the queue:



1. To display the queue:



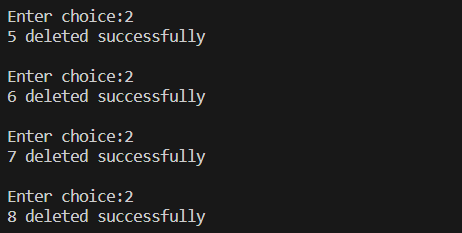
1. To peek the front element:



1. To check for queue underflow:



1. To delete elements from the queue:



1. To check for queue underflow:



1. To delete an element from an empty queue:

Expt no: 5



1. To display the queue (empty queue):



1. To peek the front element (empty queue):

