**WEEK 6 DS Stack and Queues implementation using LL:**

**Stack using linked list:**

#include <stdio.h>

#include <stdlib.h>

void push();

void pop();

void display();

struct node

{

int val;

struct node \*next;

};

struct node \*head;

void main ()

{

int choice=0;

printf("\nStack operations using linked list\n");

while(choice != 4)

{

printf("\n\nChoose one from the below options...\n");

printf("\n1.Push\n2.Pop\n3.Show\n4.Exit");

printf("\n Enter your choice \n");

scanf("%d",&choice);

switch(choice)

{

case 1:

{

push();

break;

}

case 2:

{

pop();

break;

}

case 3:

{

display();

break;

}

case 4:

{

printf("Exiting....");

break;

}

default:

{

printf("Please Enter valid choice ");

}

};

}

}

void push ()

{

int val;

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

if(ptr == NULL)

{

printf("Not able to push the element");

}

else

{

printf("Enter the value");

scanf("%d",&val);

if(head==NULL)

{

ptr->val = val;

ptr -> next = NULL;

head=ptr;

}

else

{

ptr->val = val;

ptr->next = head;

head=ptr;

}

printf("Item pushed");

}

}

void pop()

{

int item;

struct node \*ptr;

if (head == NULL)

{

printf("Underflow");

}

else

{

item = head->val;

ptr = head;

head = head->next;

free(ptr);

printf("Item popped");

}

}

void display()

{

int i;

struct node \*ptr;

ptr=head;

if(ptr == NULL)

{

printf("Stack is empty\n");

}

else

{

printf("Printing Stack elements \n");

while(ptr!=NULL)

{

printf("%d\n",ptr->val);

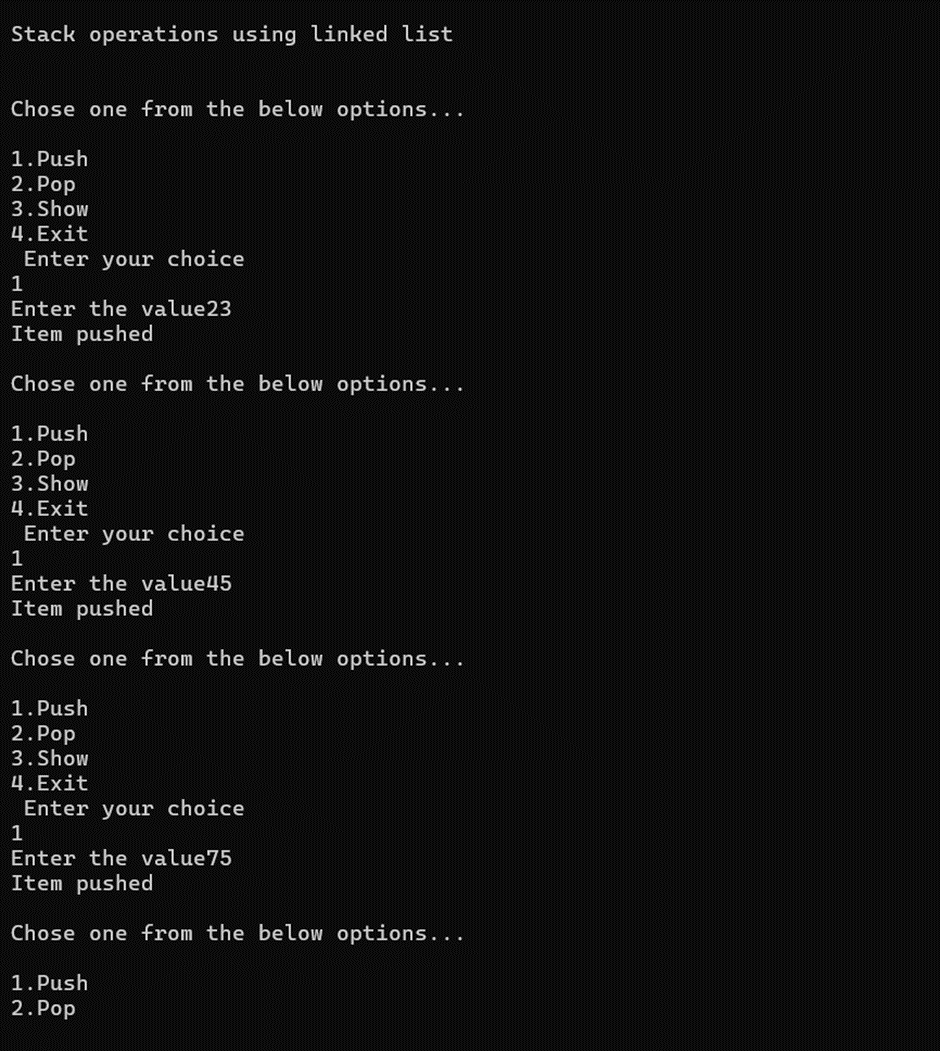
ptr = ptr->next;

}

}

}

**Output:**

****

**Queue using linked list:**

#include<stdio.h>

#include<stdlib.h>

struct node

{

int data;

struct node \*next;

};

struct node \*front;

struct node \*rear;

void insert();

void delete();

void display();

void main ()

{

int choice;

while(choice != 4)

{

printf("\nQueue operation using linked list\n");

printf("\n1.insert an element\n2.Delete an element\n3.Display the queue\n4.Exit\n");

printf("\nEnter your choice ");

scanf("%d",& choice);

switch(choice)

{

case 1: insert();

break;

case 2: delete();

break;

case 3: display();

break;

case 4: exit(0);

break;

default: printf("\nEnter valid choice??\n");

}

}

}

void insert()

{

struct node \*ptr;

int item;

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

if(ptr == NULL)

{

printf("\nOVERFLOW\n");

return;

}

else

{

printf("\nEnter value?\n");

scanf("%d",&item);

ptr -> data = item;

if(front == NULL)

{

front = ptr;

rear = ptr;

front -> next = NULL;

rear -> next = NULL;

}

else

{

rear -> next = ptr;

rear = ptr;

rear->next = NULL;

}

}

}

void delete ()

{

struct node \*ptr;

if(front == NULL)

{

printf("\nUNDERFLOW\n");

return;

}

else

{

ptr = front;

front = front -> next;

free(ptr);

}

}

void display()

{

struct node \*ptr;

ptr = front;

if(front == NULL)

{

printf("\nEmpty queue\n");

}

else

{ printf("\nprinting values .....\n");

while(ptr != NULL)

{

printf("\n%d\n",ptr -> data);

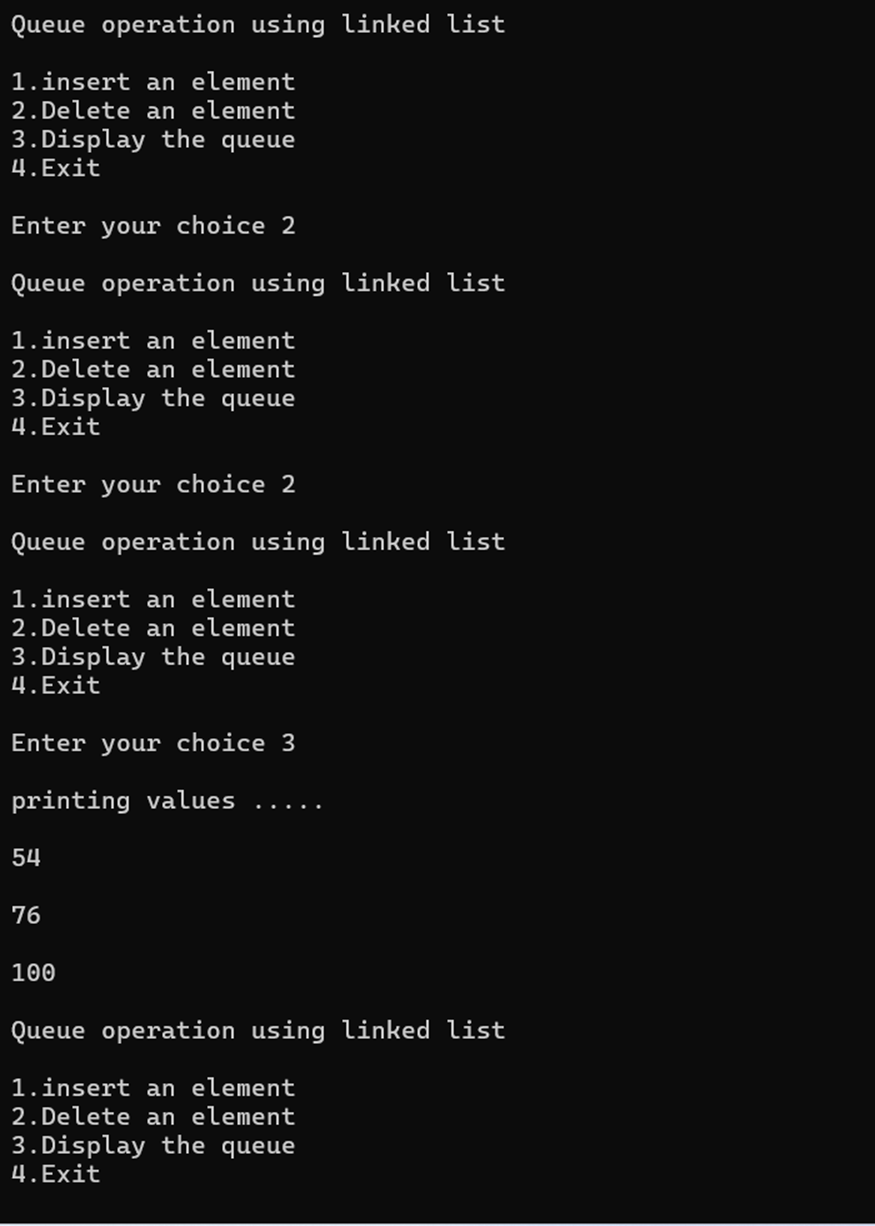
ptr = ptr -> next;

}

}

}

**Output:**

****