**DS LAB 7 : Dynamic Implementation of**

**Stack and Queue using Linked List**

Name: Vivian Vijay Ludrick

Branch: SE Comps A Batch C

RollNo: 9914

**Stack Using Linked List:**

Code:

#include <stdio.h>

#include <stdlib.h>

*// Node struct*

typedef struct node

{

int data;

struct node \*next;

} Node;

*// StackLL structure*

typedef struct

{

Node \*tos;

} StackLL;

*// push method*

void push(StackLL \**s*, int *x*)

{

Node \*p;

p = (Node \*)malloc(sizeof(Node));

p->data = *x*;

p->next = *s*->tos; *// whether it is NULL or not it doesn't matter*

*s*->tos = p;

}

*// pop method*

void pop(StackLL \**s*)

{

Node \*p;

int x;

if (*s*->tos == NULL)

{

printf("Stack underflow\n");

return;

}

else

{

p = *s*->tos;

x = p->data;

*s*->tos = p->next;

free(p);

printf("The popped element is:%d\n", x);

}

}

*// display the elements of the stack*

void display(StackLL *l*)

{

Node \*p;

p = *l*.tos;

printf("The Stack is:\n");

while (p != NULL) *// while node exists*

{

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

p = p->next;

}

}

*// main function*

int main()

{

StackLL l1;

l1.tos = NULL;

int option, x, ele;

*// menu driven*

while (1)

{

printf("Enter option: 1.Push 2.Pop 3.Display 4.Exit :\t");

scanf("%d", &option);

switch (option)

{

case 1:

printf("Enter element to be pushed :\t");

scanf("%d", &*x*);

push(&*l1*, x);

break;

case 2:

pop(&*l1*);

break;

case 3:

display(l1);

break;

case 4:

exit(0);

break;

default:

printf("Enter a valid option\n");

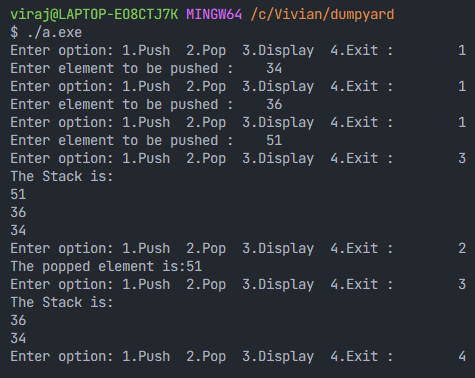
}

}

return 0;

}

OUTPUT:



**Queue Using Linked List:**

Code:

#include <stdio.h>

#include <stdlib.h>

*// Node of the linked list*

typedef struct node

{

int data;

struct node \*next;

} Node;

*// Queue using LinkedList*

typedef struct

{

Node \*front, \*rear;

} QueueLL;

*// enqueue a node*

void enqueue(QueueLL \**q*, int *x*)

{

Node \*p;

p = (Node \*)malloc(sizeof(Node)); *// memory allocation*

p->data = *x*;

p->next = NULL;

*// if it's the first node*

if (*q*->rear == NULL)

{

*q*->rear = *q*->front = p; *// both front and rear points towards p*

}

else

{

*// rear gets incremented*

*q*->rear->next = p;

*q*->rear = p;

}

}

*// dequeue a node from start*

void dequeue(QueueLL \**q*)

{

Node \*p;

int x;

if (*q*->front == NULL)

{

printf("Queue empty\n");

return;

}

else

{

p = *q*->front;

x = p->data;

*q*->front = *q*->front->next;

if (*q*->front == NULL) *// if only one node exists*

{

*q*->rear = NULL;

}

free(p); *// free the space consumed*

printf("Dequeued element is:%d\n", x);

}

}

*// display the elements of the queue*

void display(QueueLL *l*)

{

Node \*p;

printf("The linked list is:\n");

for (p = *l*.front; p != NULL; p = p->next)

{

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

}

}

*// main function*

int main()

{

QueueLL l;

l.front = NULL;

l.rear = NULL;

int option, x;

while (1)

{

printf("Enter: 1.Enqueue 2.Dequeue 3.Display 4.Exit :\t");

scanf("%d", &option);

switch (option)

{

case 1:

printf("Enter element to be enqueued :\t");

scanf("%d", &*x*);

enqueue(&*l*, x);

break;

case 2:

dequeue(&*l*);

break;

case 3:

display(l);

break;

case 4:

exit(0);

default:

printf("Enter valid option\n");

}

}

return 0;

}

OUTPUT:

