**Program to implement STACK using Linked List.**

#include<stdio.h>

#include<stdlib.h>

#include<conio.h>

void push();

int isEmpty();

void pop();

void peek();

void display();

struct node

{

int data;

struct node \*next;

};

struct node \*TOS = NULL;

int main()

{

int choice = 0, element = 0;

printf("\n===== Menu =====");

printf("\n1. PUSH");

printf("\n2. POP");

printf("\n3. PEEK");

printf("\n4. DISPLAY");

printf("\n5. Exit\n");

while (1)

{

printf("\nEnter your choice: ");

scanf("%d", &choice);

switch (choice)

{

case 1:

push();

break;

case 2:

pop();

break;

case 3:

peek();

break;

case 4:

display();

break;

case 5: return 0;

default: printf("\nEnter correct choice: ");

}

}

\_getch();

}

int isEmpty()

{

if (TOS == NULL)

return 1;

return 0;

}

void push()

{

struct node \*newNode;

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

printf("\n Enter element : ");

scanf("%d", &newNode->data);

newNode->next = NULL;

if (isEmpty())

{

TOS = newNode;

}

else

{

newNode->next = TOS;

TOS = newNode;

}

printf("Node Inserted");

}

void pop()

{

struct node \*ptr;

if (isEmpty())

{

printf("List is Empty");

return;

}

else

{

ptr = TOS;

TOS = TOS->next;

free(ptr);

}

printf("Node Deleted");

}

void peek()

{

struct node \*ptr;

if (isEmpty())

{

printf("Stack Empty");

return;

}

else

{

printf("TOP element : %d", TOS->data);

}

}

void display()

{

struct node \*ptr;

if (TOS == NULL)

{

printf("\n Stack Empty");

return;

}

ptr = TOS;

printf("\n Stack : ");

while (ptr != NULL)

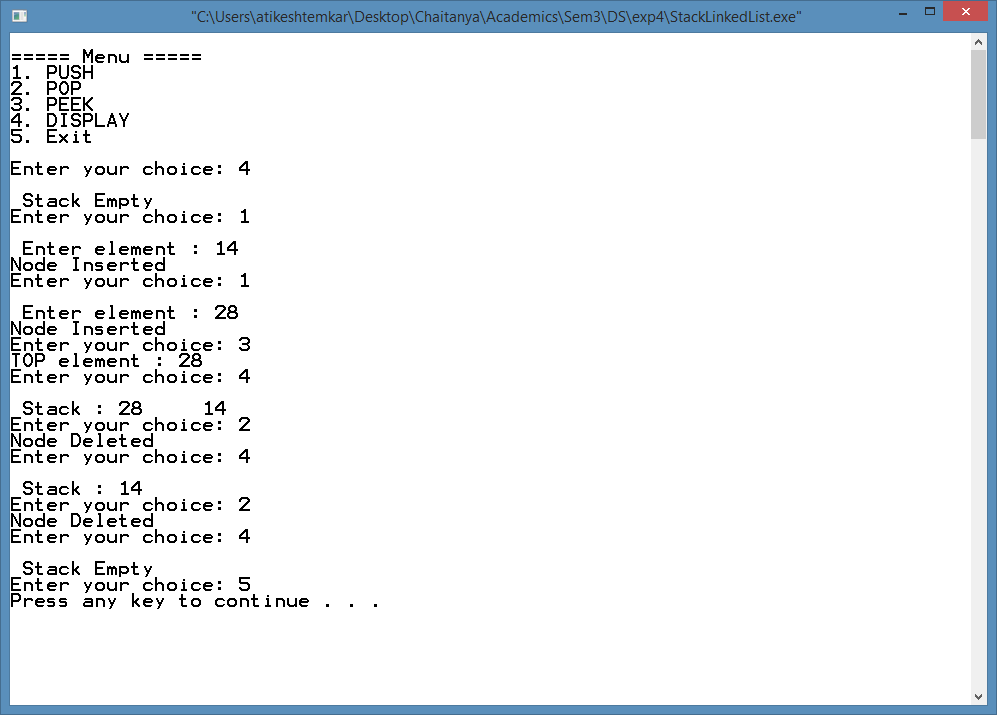
{

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

ptr = ptr->next;

}}

**Output:**



**Program to implement Queue using Linked List.**

#include<stdio.h>

#include<stdlib.h>

#include<conio.h>

void enqueue();

int isEmpty();

void dequeue();

void peek();

void display();

struct node

{

int data;

struct node \*next;

};

struct node \*front = NULL;

struct node \*rear = NULL;

int main()

{

int choice = 0, element = 0;

printf("\n ===== Menu =====");

printf("\n1. ENQUEUE");

printf("\n2. DEQUEUE");

printf("\n3. PEEK");

printf("\n4. DISPLAY");

printf("\n5. EXIT\n");

while (1)

{

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

scanf("%d", &choice);

switch (choice)

{

case 1:

enqueue();

break;

case 2:

dequeue();

break;

case 3:

peek();

break;

case 4:

display();

break;

case 5: return 0;

default: printf("\nEnter correct choice");

}

}

\_getch();

}

void enqueue()

{

struct node \*newNode, \*temp;

int element;

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

printf("\n Enter element : ");

scanf("%d", &element);

newNode->data = element;

newNode->next = NULL;

if (front == NULL)

{

front = newNode;

rear = newNode;

}

else

{

rear->next = newNode;

rear = newNode;

}

printf("Node Inserted");

}

int isEmpty()

{

if (front == NULL)

return 1;

return 0;

}

void dequeue()

{

struct node \*temp;

if (isEmpty())

{

printf("Queue Empty");

return;

}

temp = front;

front = temp->next;

free(temp);

printf("\n Element Deleted");

}

void peek()

{

if (isEmpty())

{

printf("Queue Empty");

return;

}

printf("Front element : %d", front->data);

}

void display()

{

struct node \*ptr;

if (front == NULL)

{

printf("\n Queue Empty");

return;

}

ptr = front;

printf("\n Queue :");

while (ptr != NULL)

{

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

ptr = ptr->next;

}

}

**Output:**

