STACK :

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

int n,arr[5]; int Top=-1; void Push(); void Pop(); void show();

void main()

{

int choice; clrscr();

printf("Enter size of an array:"); scanf("%d",&n);

while(1)

{

printf("\nOperations performed by Stack");

printf("\n1.Push the element\n2.Pop the element\n3.Show\n4.End"); printf("\n\nEnter the choice:");

scanf("%d",&choice);

switch(choice)

{

case 1: Push();

break; case 2: Pop();

break; case 3: show();

break; case 0: exit(0);

default: printf("\nInvalid choice!!");

}

}

getch();

}

void Push()

{ int item;

printf("value of n is : %d",n); if(Top==n-1)

{

printf("\nOverflow!!");

}

else

{

printf("\nEnter element to be inserted to the stack:"); scanf("%d",&item);

Top=Top+1; arr[Top]=item;

printf("\n Element inserted successfully...\n");

}

}

void Pop()

{

if(Top==-1)

{

printf("\nUnderflow!!");

}

else

{

printf("\nPopped element: %d",arr[Top]); Top=Top-1;

}

}

void show()

{

int i;

if(Top==-1)

{

printf("\nUnderflow!!");

}

else

{

printf("\nElements present in the stack: \n"); for(i=Top;i>=0;i--)

printf("%d\n",arr[i]);

}

}

SIMPLE QUEUE :

#include<conio.h>

#include<stdio.h> int queue[5];

int i,n;

int front=-1,rear=-1; void insert();

void deleted(); void display(); void main()

{

int i, ch;

printf("Enter the size:"); scanf("%d",&n);

do

{

printf("\nEnter your choice:\n 1.INSERT\n2.DELETE\n3.DISPLAY\n0.EXIT\n"); scanf("%d",&ch);

switch(ch)

{

case 1:

insert(); break;

case 2:

deleted(); break;

case 3:

display(); break;

default:

printf("enter valid option..."); break;

}

}while(ch!=0); getch();

}

void insert()

{

int item;

if(rear>=n-1)

{

printf("queue is overflow");

}

else

{

printf("Enter element to be inserted"); scanf("%d",&item);

rear=rear+1;

queue[rear]=item; if(front==-1)

{

front=0;

}

printf("Insertion done... & item inserted=%d",item);

}

}

void deleted()

{

int item;

if(front==-1)

{

printf("queue is underflow");

}

else

{

printf("Item deleted...%d",queue[front]); if(front==rear)

{

front=-1; rear=-1;

}

else

{

front=front+1;

}

}

}

void display()

{

int i;

if(front==-1)

{

printf("queue is empty");

}

else

{

for(i=front;i<=rear;i++)

{

printf("%d\n",queue[i]);

}

}

}

CIRCULAR QUEUE

#include<stdio.h> # define MAX 5

int cqueue\_arr[MAX]; int front = -1;

int rear = -1;

void insert(int item)

{

if((front == 0 && rear == MAX-1) || (front == rear+1))

{

printf("Queue Overflow "); return;

}

if(front == -1)

{

front = 0;

rear = 0;

}

else

{

if(rear == MAX-1) rear = 0;

else

rear = rear+1;

}

cqueue\_arr[rear] = item ;

}

void deletion()

{

if(front == -1)

{

printf("Queue Underflown"); return ;

}

printf("Element deleted from queue is : %dn",cqueue\_arr[front]); if(front == rear)

{

}

else

{

}

front = -1; rear=-1;

if(front == MAX-1) front = 0;

else

front = front+1;

}

void display()

{

int front\_pos = front,rear\_pos = rear; if(front == -1)

{

printf("Queue is emptyn"); return;

}

printf("Queue elements :n"); if( front\_pos <= rear\_pos ) while(front\_pos <= rear\_pos)

{

}

else

{

printf("%d ",cqueue\_arr[front\_pos]); front\_pos++;

while(front\_pos <= MAX-1)

{

printf("%d ",cqueue\_arr[front\_pos]); front\_pos++;

}

front\_pos = 0;

while(front\_pos <= rear\_pos)

{

printf("%d ",cqueue\_arr[front\_pos]); front\_pos++;

}

}

printf("n");

}

int main()

{

int choice,item; do

{

printf("1.Insertn"); printf("2.Deleten"); printf("3.Displayn"); printf("4.Quitn");

printf("Enter your choice : "); scanf("%d",&choice); switch(choice)

{

case 1 :

printf("Input the element for insertion in queue : "); scanf("%d", &item);

insert(item); break;

case 2 :

deletion(); break; case 3:

display(); break; case 4:

break; default:

printf("Wrong choicen");

}

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

}

DQUEUE:

#include <stdio.h> #include<conio.h> #define size 5

int deque[size]; int f = -1, r = -1;

// insert\_front function will insert the value from the front void insert\_front(int x)

{

if((f==0 && r==size-1))

{

printf("Overflow");

}

else if((f==-1) && (r==-1))

{

f=r=0;

deque[f]=x;

}

else

{

f=f-1;

deque[f]=x;

}

}

// insert\_rear function will insert the value from the rear void insert\_rear(int x)

{

if((f==0 && r==size-1))

{

printf("Overflow");

}

else if((f==-1) && (r==-1))

{

r=0;

deque[r]=x;

}

else

{

r++;

deque[r]=x;

}

}

// display function prints all the value of deque. void display()

{

int i=f;

printf("\nElements in a deque are: ");

while(i<=r)

{

printf("%d ",deque[i]); i++;

}

}

// getfront function retrieves the first value of the deque. void getfront()

{

if((f==-1) && (r==-1))

{

printf("Deque is empty");

}

else

{

printf("\nThe value of the element at front is: %d", deque[f]);

}

}

// getrear function retrieves the last value of the deque. void getrear()

{

if((f==-1) && (r==-1))

{

printf("Deque is empty");

}

else

{

printf("\nThe value of the element at rear is %d", deque[r]);

}

}

// delete\_front() function deletes the element from the front void delete\_front()

{

if((f==-1) && (r==-1))

{

printf("Deque is empty");

}

else if(f==r)

{

printf("\nThe deleted element is %d", deque[f]); f=-1;

r=-1;

}

else

{

}

}

printf("\nThe deleted element is %d", deque[f]); f=f+1;

// delete\_rear() function deletes the element from the rear void delete\_rear()

{

if((f==-1) && (r==-1))

{

printf("Deque is empty");

}

else if(f==r)

{

printf("\nThe deleted element is %d", deque[r]); f=-1;

r=-1;

}

else

{

}

}

printf("\nThe deleted element is %d", deque[r]); r=r-1;

void main()

{

insert\_front(20); insert\_front(10); insert\_rear(30); insert\_rear(50); insert\_rear(80);

display(); // Calling the display function to retrieve the values of deque

getfront(); // Retrieve the value at front-end getrear(); // Retrieve the value at rear-end delete\_front();

delete\_rear();

display(); // calling display function to retrieve values after deletion getch();

}

LINKEDLIST

#include<stdio.h> #include<stdlib.h> struct node

{

int data;

struct node \*next;

};

struct node \*head;

void beginsert (); void lastinsert ();

void randominsert(); void begin\_delete(); void last\_delete();

void random\_delete(); void display();

void search(); void main ()

{

int choice =0;

while(choice != 9)

{

printf("\n\n\*\*\*\*\*\*\*\*\*Main Menu\*\*\*\*\*\*\*\*\*\n");

printf("\nChoose one option from the following list ...\n");

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

printf("\n1.Insert in begining\n2.Insert at last\n3.Insert at any random location\n4.Delete from Beginning\n

5.Delete from last\n6.Delete node after specified

location\n7.Search for an element\n8.Show\n9.Exit\n"); printf("\nEnter your choice?\n");

scanf("\n%d",&choice);

switch(choice)

{

case 1:

beginsert(); break;

case 2:

lastinsert(); break;

case 3:

randominsert(); break;

case 4:

begin\_delete(); break;

case 5:

last\_delete(); break;

case 6:

random\_delete(); break;

case 7:

search(); break; case 8:

display();

break; case 9:

exit(0); break; default:

printf("Please enter valid choice..");

}

}

}

void beginsert()

{

struct node \*ptr; int item;

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

{

printf("\nOVERFLOW");

}

else

{

printf("\nEnter value\n"); scanf("%d",&item);

ptr->data = item; ptr->next = head; head = ptr;

printf("\nNode inserted");

}

}

void lastinsert()

{

struct node \*ptr,\*temp; int item;

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

{

printf("\nOVERFLOW");

}

else

{

printf("\nEnter value?\n"); scanf("%d",&item);

ptr->data = item; if(head == NULL)

{

ptr -> next = NULL; head = ptr;

printf("\nNode inserted");

}

else

{

temp = head;

while (temp -> next != NULL)

{

temp = temp -> next;

}

temp->next = ptr; ptr->next = NULL;

printf("\nNode inserted");

}

}

}

void randominsert()

{

int i,loc,item;

struct node \*ptr, \*temp;

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

{

printf("\nOVERFLOW");

}

else

{

printf("\nEnter element value");

scanf("%d",&item); ptr->data = item;

printf("\nEnter the location after which you want to insert "); scanf("\n%d",&loc);

temp=head;

for(i=0;i<loc;i++)

{

temp = temp->next; if(temp == NULL)

{

printf("\ncan't insert\n"); return;

}

}

ptr ->next = temp ->next; temp ->next = ptr;

printf("\nNode inserted");

}

}

void begin\_delete()

{

struct node \*ptr; if(head == NULL)

{

printf("\nList is empty\n");

}

else

{

ptr = head;

head = ptr->next; free(ptr);

printf("\nNode deleted from the begining ...\n");

}

}

void last\_delete()

{

struct node \*ptr,\*ptr1; if(head == NULL)

{

printf("\nlist is empty");

}

else if(head -> next == NULL)

{

head = NULL; free(head);

printf("\nOnly node of the list deleted ...\n");

}

else

{

ptr = head;

while(ptr->next != NULL)

{

ptr1 = ptr;

ptr = ptr ->next;

}

ptr1->next = NULL; free(ptr);

printf("\nDeleted Node from the last ...\n");

}

}

void random\_delete()

{

struct node \*ptr,\*ptr1; int loc,i;

printf("\n Enter the location of the node after which you want to perform deletion \n");

scanf("%d",&loc); 2 ptr=head;

for(i=0;i<loc;i++)

{

ptr1 = ptr;

ptr = ptr->next;

if(ptr == NULL)

{

printf("\nCan't delete"); return;

}

}

ptr1 ->next = ptr ->next; free(ptr);

printf("\nDeleted node %d ",loc+1);

}

void search()

{

struct node \*ptr; int item,i=0,flag; ptr = head;

if(ptr == NULL)

{

printf("\nEmpty List\n");

}

else

{

printf("\nEnter item which you want to search?\n"); scanf("%d",&item);

while (ptr!=NULL)

{

if(ptr->data == item)

{

printf("item found at location %d ",i+1); flag=0;

}

else

{

flag=1;

}

i++;

ptr = ptr -> next;

}

if(flag==1)

{

printf("Item not found\n");

}

}

}

void display()

{

struct node \*ptr; ptr = head;

if(ptr == NULL)

{

printf("Nothing to print");

}

else

{

printf("\nprinting values \n");

while (ptr!=NULL)

{

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

}

}

}