Practical-5

Implementing different types of Queue and its Operations

1. Write a C program to implement all operations of normal queue using array.

* Code:

#include<stdio.h>

#include<stdlib.h>

#define size 5

void insert();

void delete();

void display();

int queue\_arr[size];

int rear=-1,front=-1;

int main()

{

int c;

while(1)

{

printf("\n1.Insert element to queue");

printf("\n2.Delete element from queue");

printf("\n3.Display all elements of queue");

printf("\n4.Exit");

printf("\nEnter your choice : ");

scanf("%d", &c);

switch(c)

{

case 1:insert();

break;

case 2:delete();

break;

case 3:display();

break;

case 4:exit(1);

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

break;

}

}

return 0;

}

void insert()

{

int add;

if(rear==size-1)

printf("Queue overflow\n");

else

{

if(front==-1)

front=0;

printf("\nEnter element to insert into queue : ");

scanf("%d", &add);

rear++;

queue\_arr[rear]=add;

}

}

void delete()

{

if(front==-1)

printf("Queue underflow\n");

else

{

printf("\nElement deleted from queue is : %d\n", queue\_arr[front]);

front++;

}

}

void display()

{

int i;

if(front==-1)

printf("Queue is empty\n");

printf("\n");

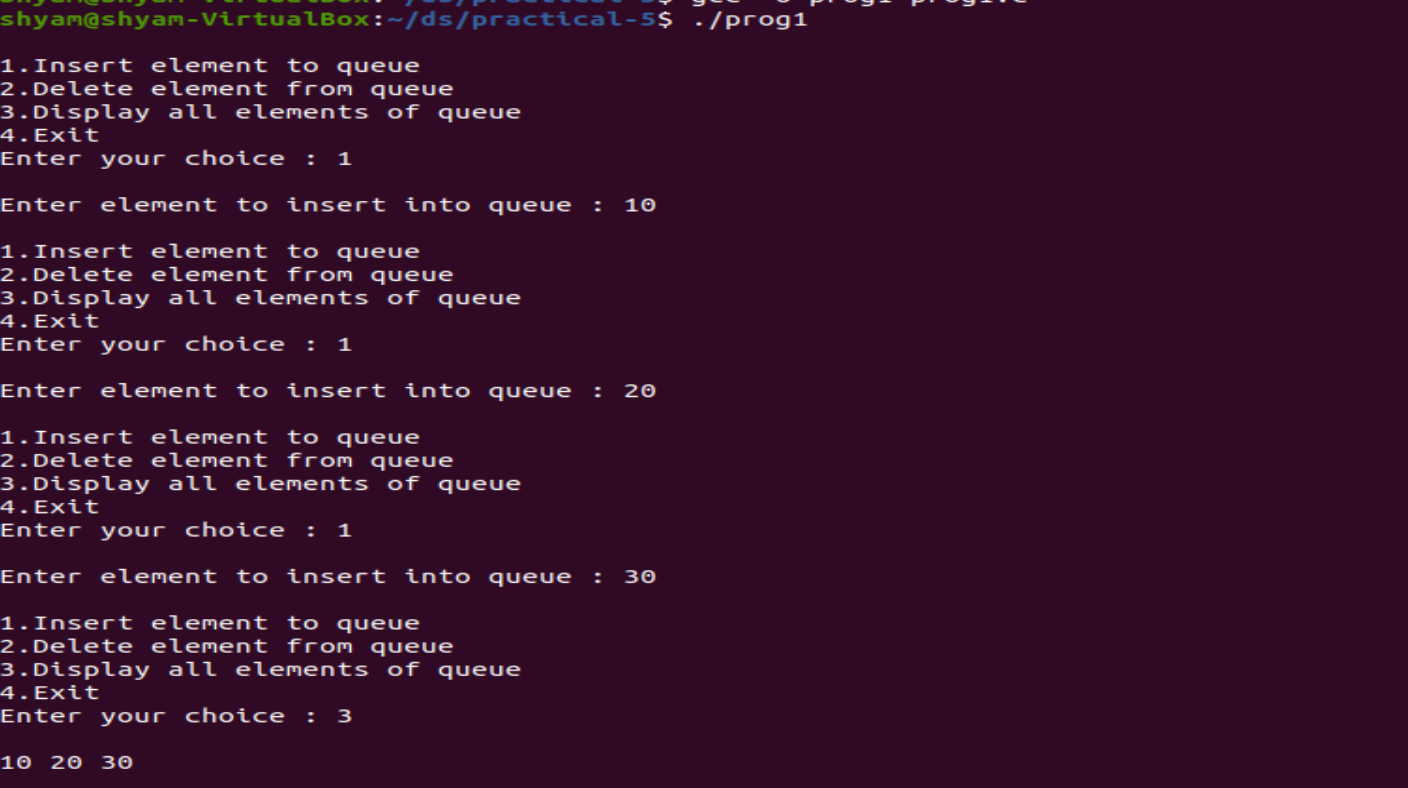
for(i=front;i<=rear;i++)

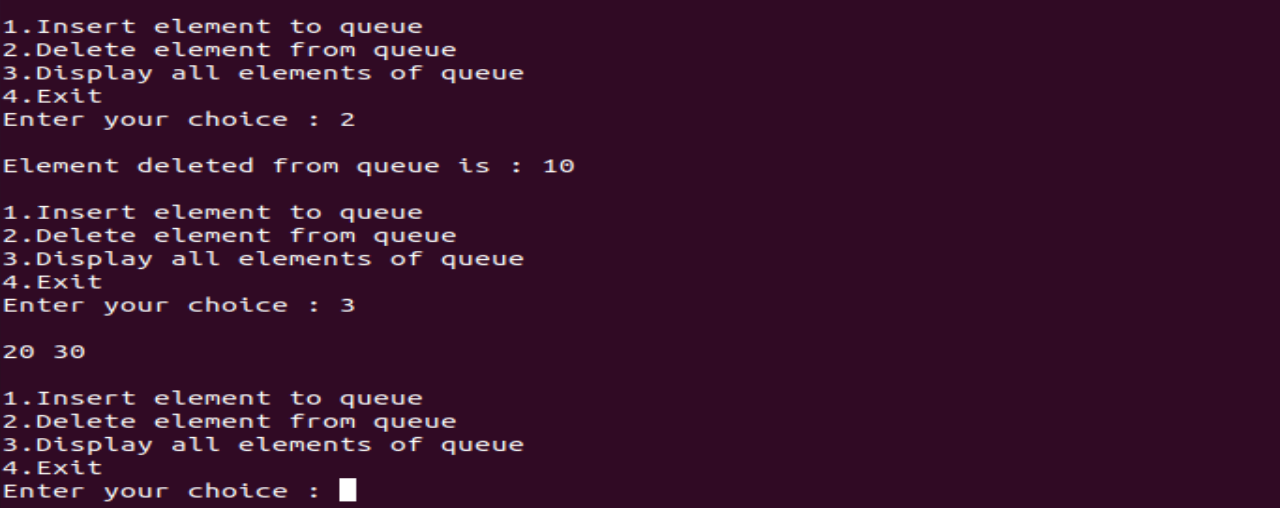
printf("%d ",queue\_arr[i]);

printf("\n");

}

* Output





1. Write a C program to implement all operations of circular queue using array.

* Code:

#include<stdio.h>

#include<stdlib.h>

#define max 5

int cqueue\_arr[max];

int front=-1,rear=-1;

void insert(int item)

{

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

{

printf("Queue overflow\n");

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 underflow\n");

return;

}

printf("Element deleted from queue is : %d\n", cqueue\_arr[front]);

if(front==rear)

{

front=-1;

rear=-1;

}

else

{

if(front==max-1)

front=0;

else

front=front+1;

}

}

void display(int Q[])

{

if(front==-1 && rear==-1)

{

printf("Queue is empty\n");

return;

}

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

{

if(i==rear)

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

else

printf("%d-->", Q[i]);

}

}

int main()

{

int choice,item;

do

{

printf("1.Insert\n");

printf("2.Delete\n");

printf("3.Display\n");

printf("4.Exit\n");

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(cqueue\_arr);

break;

case 4: exit(0);

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

}

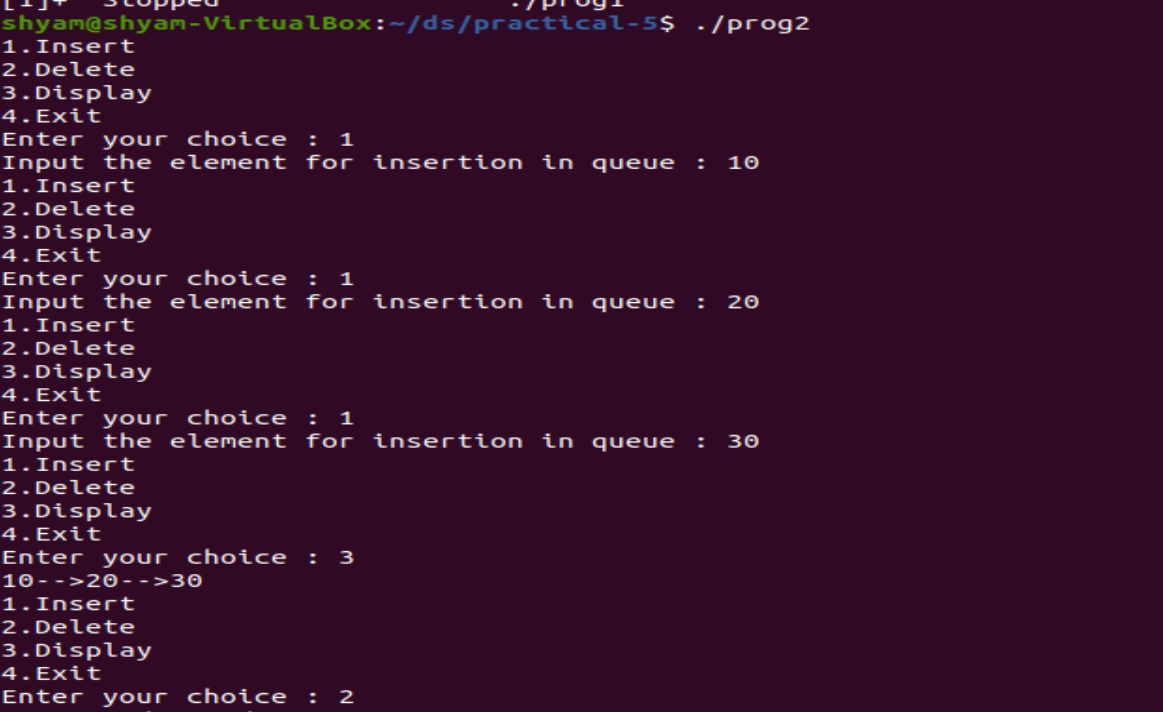
}

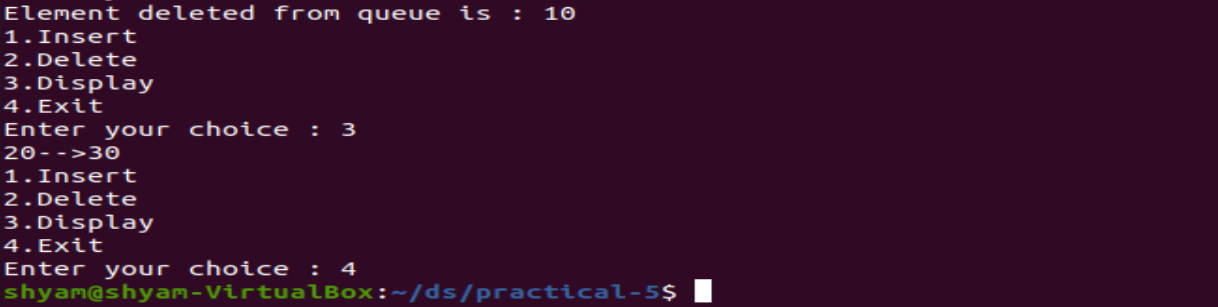
while(choice!=4);

return 0;

}

* Output:





1. Write a C program to create a queue where insertion and deletion of elements can be done from both the ends.

* Code:

#include<stdio.h>

#include<stdlib.h>

#define max 7

int deque\_arr[max];

int front=-1;

int rear=-1;

void insert\_frontend(int item);

void insert\_rearend(int item);

int delete\_frontend();

int delete\_rearend();

void display();

int isEmpty();

int isFull();

int main()

{

int choice,item;

do

{

printf("\n\n1.Insert at the front end\n");

printf("2.Insert at the rear end\n");

printf("3.Delete from front end\n");

printf("4.Delete from rear end\n");

printf("5.Display\n");

printf("\nEnter your choice : ");

scanf("%d", &choice);

switch(choice)

{

case 1:printf("\nInput the element for adding in queue : ");

scanf("%d", &item);

insert\_frontend(item);

break;

case 2:printf("\nInput the element for adding in queue : ");

scanf("%d", &item);

insert\_rearend(item);

break;

case 3:printf("\nElement deleted from front end is : %d\n", delete\_frontend());

break;

case 4:printf("\nElement deleted from rear end is : %d\n", delete\_rearend());

break;

case 5:display();

break;

case 6:exit(1);

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

}

printf("\nFront = %d, rear=%d\n", front,rear);

display();

}while(choice!=6);

}

void insert\_frontend(int item)

{

if(isFull())

{

printf("\nQueue overflow\n");

return;

}

if(front==-1)

{

front=0;

rear=0;

}

else if(front==0)

front=max-1;

else

front=front-1;

deque\_arr[front]=item;

}

void insert\_rearend(int item)

{

if(isFull())

{

printf("\nQueue overflow\n");

return;

}

if(front==-1)

{

front=0;

rear=0;

}

else if(rear==max-1)

rear=0;

else

rear=rear+1;

deque\_arr[rear]=item;

}

int delete\_frontend()

{

int item;

if(isEmpty())

{

printf("\nQueue underflow\n");

exit(1);

}

item=deque\_arr[front];

if(front==rear)

{

front=-1;

rear=-1;

}

else

if(front==max-1)

front=0;

else

front=front+1;

return item;

}

int delete\_rearend()

{

int item;

if(isEmpty())

{

printf("\nQueue underflow\n");

exit(1);

}

item=deque\_arr[rear];

if(front==rear)

{

front=-1;

rear=-1;

}

else

if(rear==0)

rear=max-1;

else

rear=rear-1;

return item;

}

int isFull()

{

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

return 1;

else

return 0;

}

int isEmpty()

{

if(front==-1)

return 1;

else

return 0;

}

void display()

{

int i;

if(isEmpty())

{

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

return;

}

printf("\nQueue elements : \n");

i=front;

if(front<=rear)

{

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

{

if(i==front)

printf("\n%d-->front", deque\_arr[i]);

else if(i==rear)

printf("\n%d--->rear",deque\_arr[i]);

else

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

}

}

else

{

while(i<=max-1)

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

i=0;

while(i<=rear)

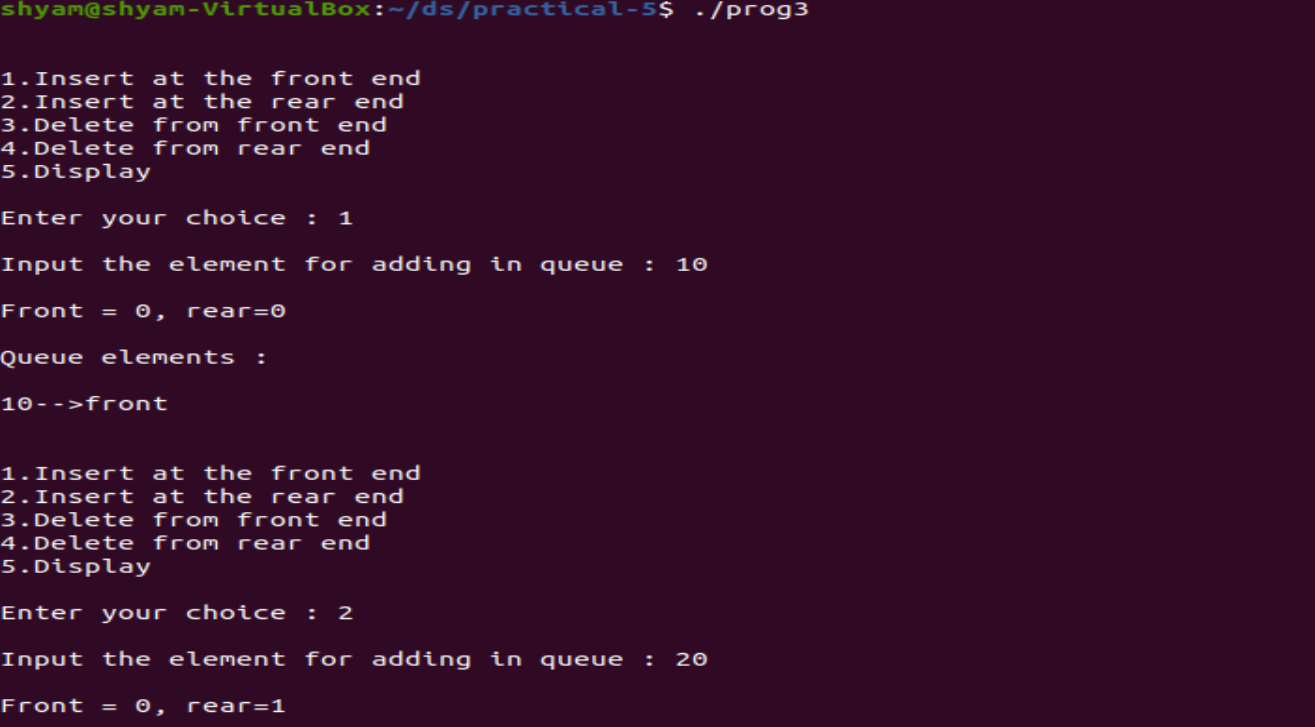
printf("%d ", deque\_arr[i++]);

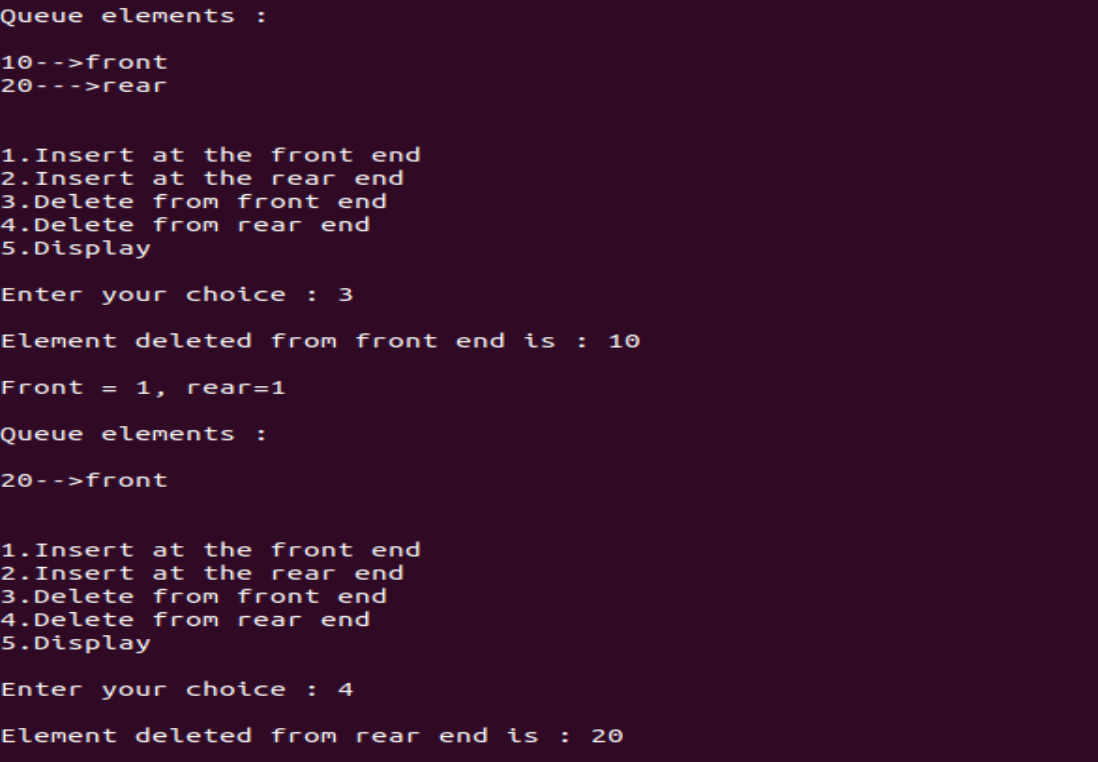
}

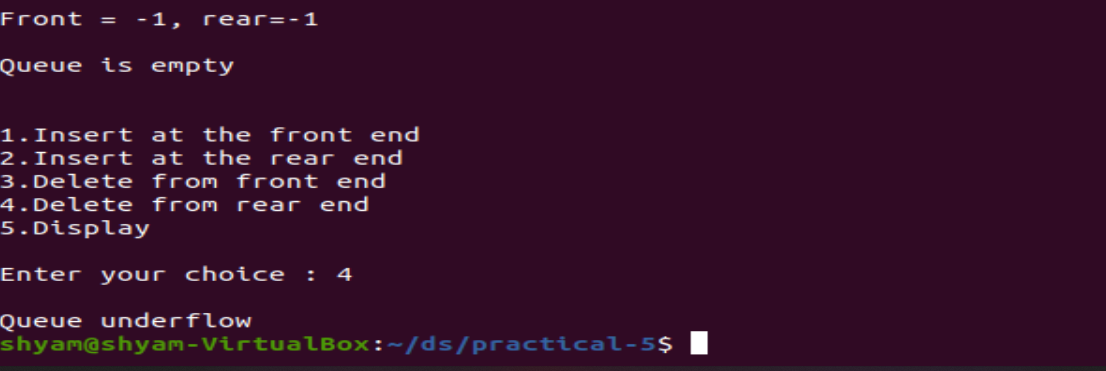
printf("\n");

}

* Output:







1. Write a C program to reverse a normal queue.

* Code:

#include<stdio.h>

#include<stdlib.h>

void insert();

void print();

void reverse();

int front=-1,rear=-1;

int queue\_arr[50];

void main()

{

int n,i=0,t;

printf("Enter the size of queue : ");

scanf("%d", &n);

printf("\nEnter the data for queue\n");

while(i<n)

{

scanf("%d", &t);

insert();

i++;

}

printf("\nQueue which you have entered : ");

print();

reverse();

printf("\nQueue after reversing : ");

print();

}

void insert()

{

int add;

if(rear==50-1)

printf("Queue overflow\n");

else

{

if(front==-1)

front=0;

printf("Enter element to insert in the queue : ");

scanf("%d", &add);

rear=rear+1;

queue\_arr[rear]=add;

}

}

void print()

{

int i;

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

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

}

void reverse()

{

int i,j,t;

for(i=front,j=rear;i<j;i++,j--)

{

t=queue\_arr[i];

queue\_arr[i]=queue\_arr[j];

queue\_arr[j]=t;

}

}

* Output:

