Program 1: Implementation of queues using arrays

#include<stdio.h>

#include<stdlib.h>

#define MAX 5

int q[MAX];

int front=-1;

int rear=-1;

int isfull();

int isempty();

void insert(int x);

int del();

int peek();

void display();

int main()

{

int c,x;

start:

do

{

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

printf("1.Insert 2.Delete 3.Display front 4.Display all 0.Exit\n");

scanf("%d",&c);

switch(c)

{

case 1:printf("\nEnter element to be inserted\n");

scanf("%d",&x);

insert(x);

printf("\n");

break;

case 2:x=del();

if(x!= -1)

printf("%d was deleted\n",x);

else

goto start;

break;

case 3:x=peek();

if(x!= -1)

printf("Element at the front is %d\n",x);

else

goto start;

break;

case 4:display();

printf("\n");

break;

default:if(c)

printf("Invalid option. Select again\n");

}

}while(c!=0);

}

void insert(int x)

{

if(isfull())

printf("Queue overflow\n");

else

{

q[++rear]=x;

if(front == -1)

front=0;

}

}

int del()

{

if(isempty())

{

printf("Queue underflow\n");

return -1;

}

else

{

front++;

return q[front-1];

}

}

int peek()

{

if(isempty())

{

printf("Queue underflow\n");

return -1;

}

else

return q[front];

}

int isempty()

{

if(rear ==-1 || front == rear+1)

{

if(front == rear+1)

front=rear=-1;

return 1;

}

else

return 0;

}

int isfull()

{

if(rear == MAX-1)

return 1;

else

return 0;

}

void display()

{

if(isempty())

printf("Empty queue\n");

else

{

printf("The elements in the queue are:\n");

int i;

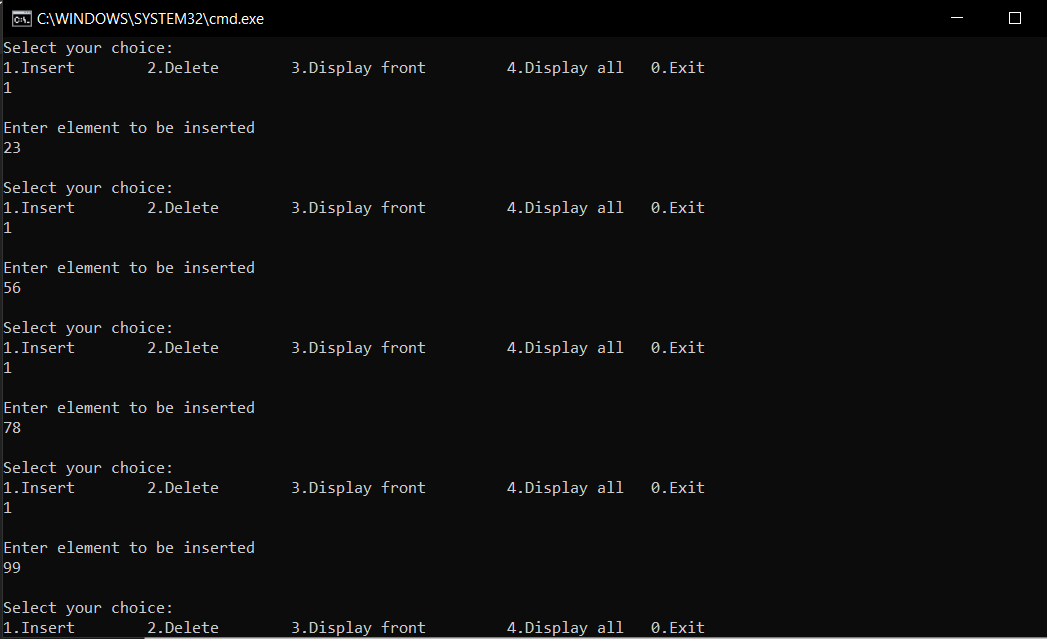
for(i=front;i<=rear;i++)

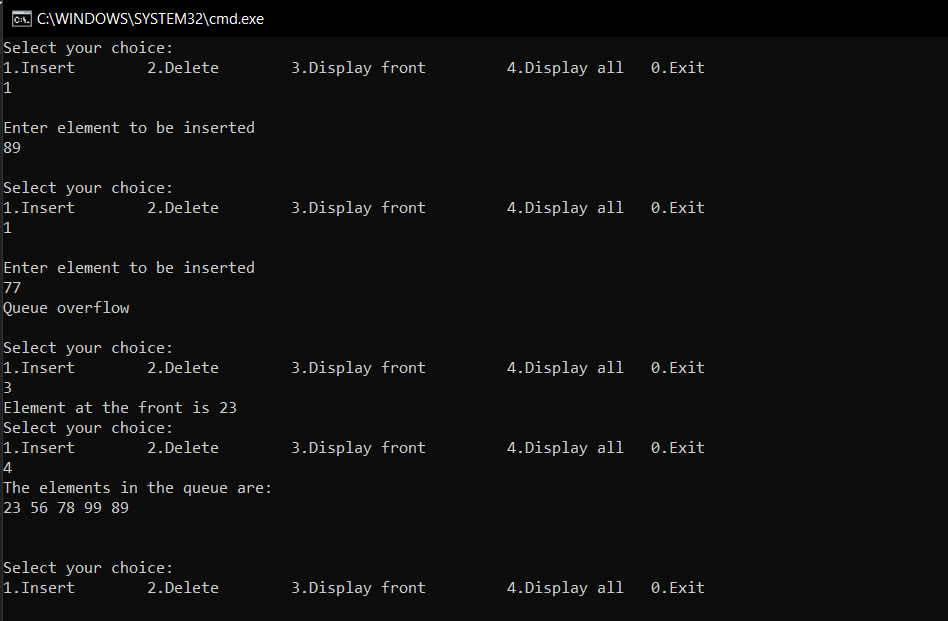
printf("%d ",q[i]);

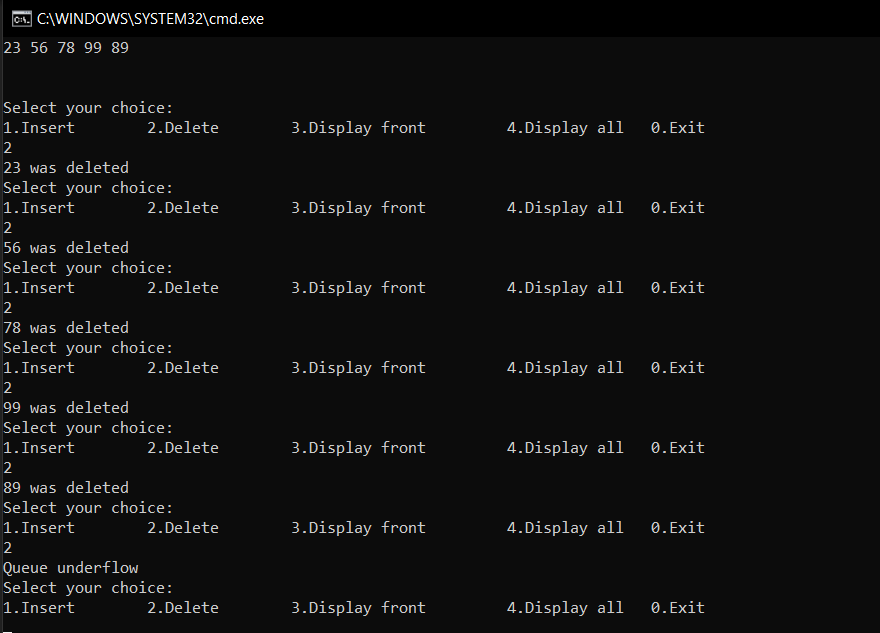
printf("\n\n");

}

}//output : enqueue, display and dequeue until empty







Program 2: Implementation of queues using linked lists

#include<stdio.h>

#include<stdlib.h>

struct element

{

int data;

struct element\*next;

};

typedef struct element node;

node \* front= NULL;

node \* rear= NULL;

int isfull();

int isempty();

void insert(int x);

int del();

int peek();

void display();

int main()

{

int c,x;

start:

do

{

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

printf("1.Insert 2.Delete 3.Display front 4.Display all 0.Exit\n");

scanf("%d",&c);

switch(c)

{

case 1:printf("\nEnter element to be inserted\n");

scanf("%d",&x);

insert(x);

printf("\n\n");

break;

case 2:x=del();

if(x!= -1)

printf("%d was deleted\n\n",x);

else

goto start;

break;

case 3:x=peek();

if(x!= -1)

printf("Element at the front is %d\n\n",x);

else

goto start;

break;

case 4:display();

printf("\n\n");

break;

default:if(c)

printf("Invalid option. Select again\n\n");

}

}while(c!=0);

}

void insert(int x)

{

node \*temp = (node\*)malloc(sizeof(node));

temp->next=NULL;

temp->data=x;

if(front ==NULL && rear == NULL)

front=rear=temp;

else

{

rear->next=temp;

rear=temp;

}

}

int del()

{

if(isempty())

{

printf("Queue underflow\n\n\n");

return -1;

}

else

{

node \*temp=front; int x;

front = front->next;

x=temp->data;

free(temp);

if(front == NULL)

rear = NULL;

return x;

}

}

int peek()

{

if(isempty())

{

printf("Queue underflow\n\n\n");

return -1;

}

else

return front->data;

}

int isempty()

{

if(front == NULL && rear == NULL)

return 1;

else

return 0;

}

void display()

{

if(isempty())

printf("Empty Queue\n");

else

{

node \*ptr = front;

printf("The elements in the queue are as follows:\n");

while(ptr!=NULL)

{

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

ptr=ptr->next;

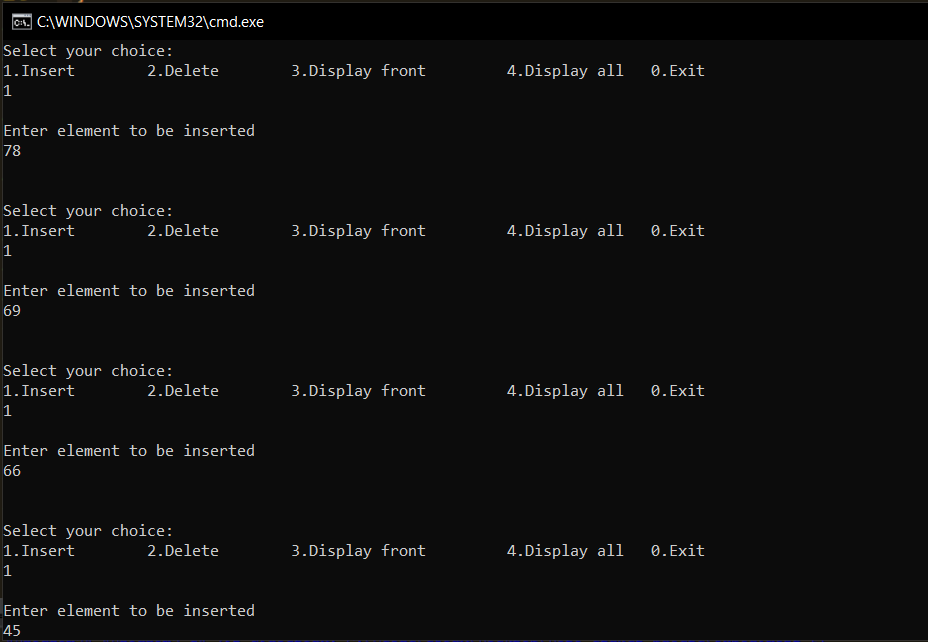
}

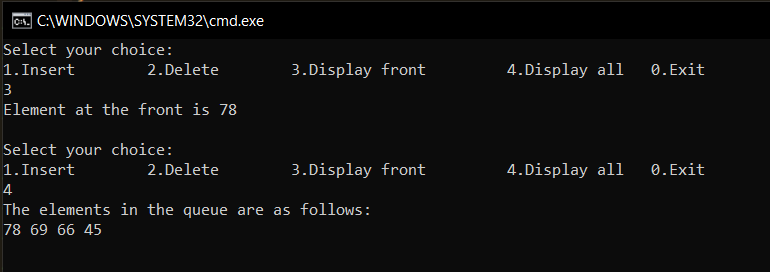
printf("\n\n");

}

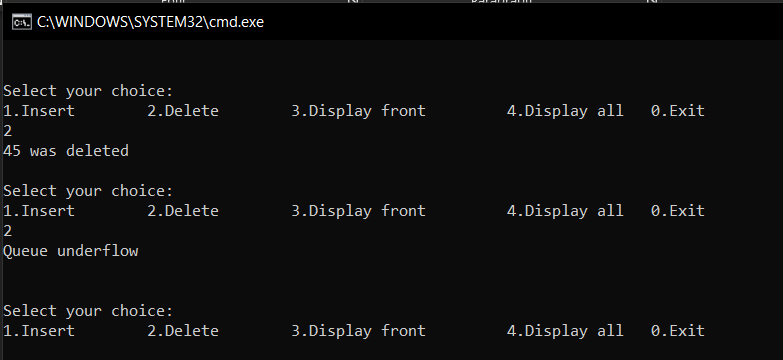
}

//output: Insert, display and delete till empty









Program 3: Implementation of circular queues using linked lists

#include<stdio.h>

#include<stdlib.h>

struct element

{

int data;

struct element\*next;

};

typedef struct element node;

node \* rear = NULL;

node \*front =NULL;

int isempty();

void insert(int x);

int del();

void display();

int main()

{

int c,x;

start:

do

{

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

printf("1.Insert 2.Delete 3.Display all 0.Exit\n");

scanf("%d",&c);

switch(c)

{

case 1:printf("\nEnter element to be inserted\n");

scanf("%d",&x);

insert(x);

printf("\n\n");

break;

case 2:x=del();

if(x!= -1)

printf("%d was deleted\n\n",x);

else

goto start;

break;

case 3:display();

printf("\n\n");

break;

default:if(c)

printf("Invalid option. Select again\n\n");

}

}while(c!=0);

}

void insert(int x)

{

node \*temp = (node\*)malloc(sizeof(node));

temp->next=NULL;

temp->data=x;

if(rear == NULL)

{

front = rear = temp;

rear->next = front;

}

else

{

rear->next = temp;

temp->next = front;

rear = temp;

}

}int del()

{

if(isempty())

{

printf("Queue underflow\n\n\n");

return -1;

}

else

{

node \*temp=front; int x;

front = front->next;

x=temp->data;

free(temp);

if(front == rear->next)

{

printf("%d was deleted\n\n",x);

front = rear = NULL;

return -1;

}

return x;

}

}

int isempty()

{

if(front == NULL && rear == NULL)

return 1;

else

return 0;

}

void display()

{

if(isempty())

printf("Empty Queue\n");

else

{

node \*ptr = front;

printf("The elements in the queue are as follows:\n");

do

{

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

ptr=ptr->next;

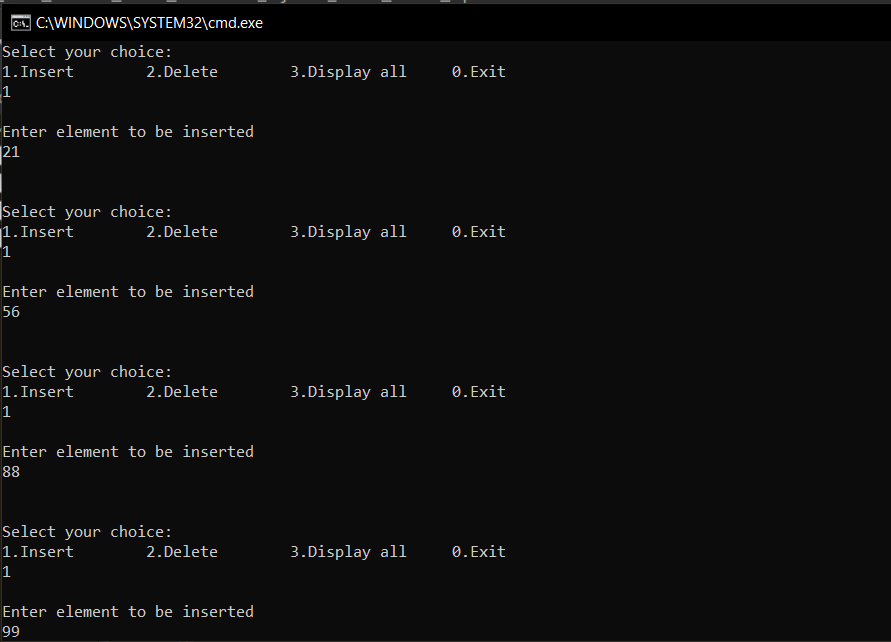
}while(ptr!=rear->next);

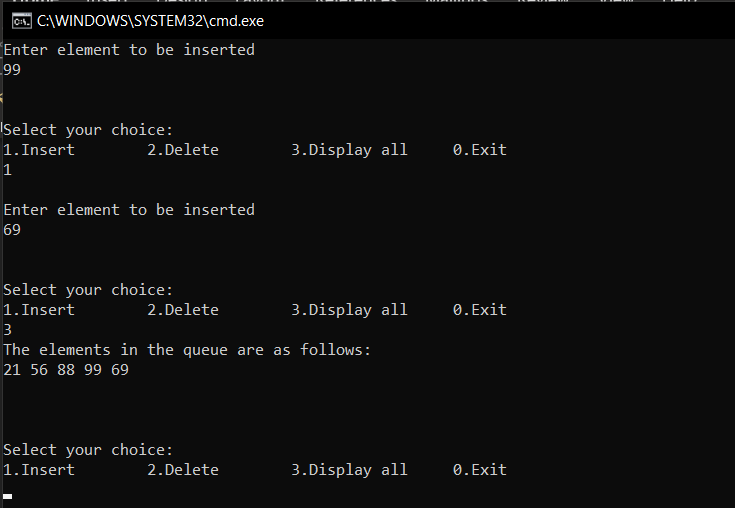
printf("\n\n");

}

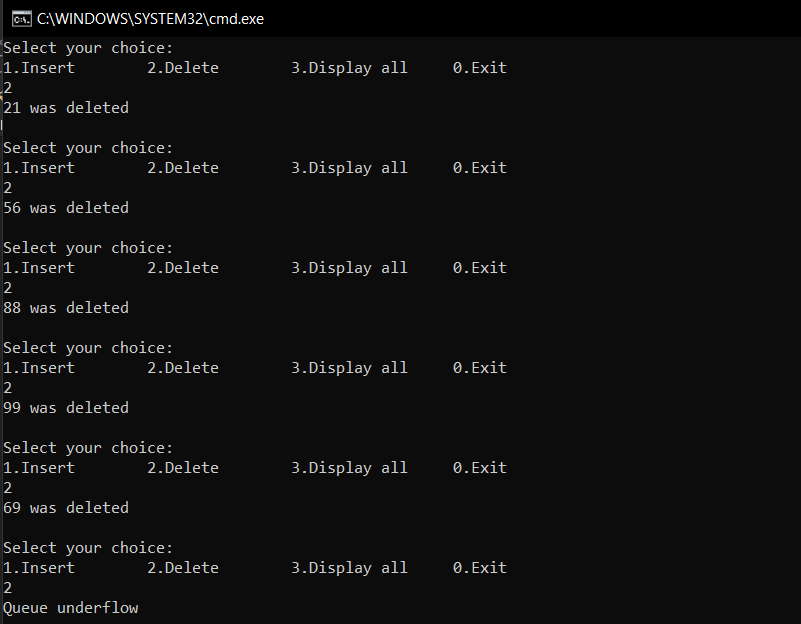
}

Output: Enqueue and display





Output: Dequeue till empty



Program 4: Implementation of circular queue using arrays

#include<stdio.h>

#include<stdlib.h>

#define MAX 5

int q[MAX];

int front=-1;

int rear=-1;

void insert(int x);

int del();

void display();

int main()

{

int c,x;

start:

do

{

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

printf("1.Insert 2.Delete 3.Display all 0.Exit\n");

scanf("%d",&c);

switch(c)

{

case 1:printf("\nEnter element to be inserted\n");

scanf("%d",&x);

insert(x);

printf("\n");

break;

case 2:x=del();

if(x!= -1)

printf("%d was deleted\n\n",x);

else

goto start;

break;

case 3:display();

printf("\n");

break;

default:if(c)

printf("Invalid option. Select again\n\n");

}

}while(c!=0);

}

void insert(int x)

{

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

{

printf("Queue overflow\n\n");

return;

}

if(rear==MAX-1 && front!=0)

rear=-1;

if(front==-1)

front++;

q[++rear]=x;

}

int del()

{

if(front==-1)

{

printf("Queue empty\n\n");

return -1;

}

if (front == rear)

{

int x;

x=q[front];

front=rear=-1;

return x;

}

if(front == MAX-1)

{

front=0;

return q[MAX-1];

}

front++;

return q[front-1];

}

void display()

{

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

printf("Empty queue\n\n");

else

{

printf("The elements in the queue are:\n");

if(rear<front)

{

int i;

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

printf("%d ",q[i]);

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

printf("%d ",q[i]);

printf("\n\n");

}

else

{

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

printf("%d ",q[i]);

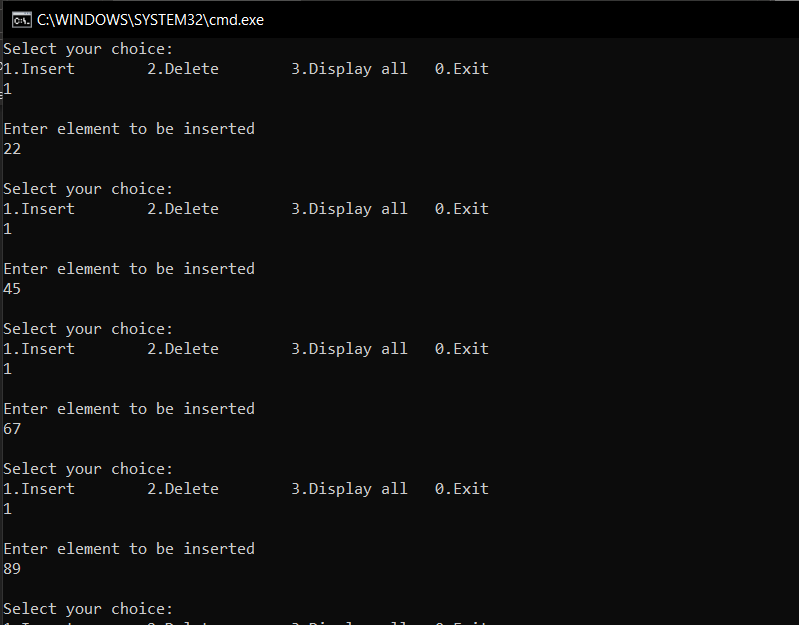
printf("\n\n");

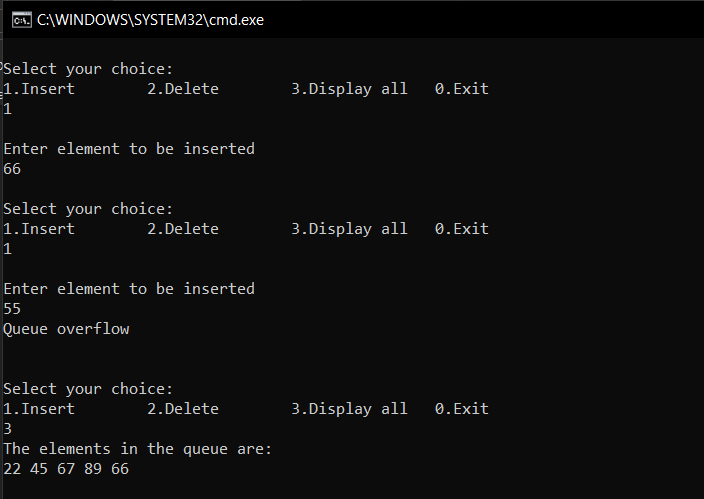
}

}

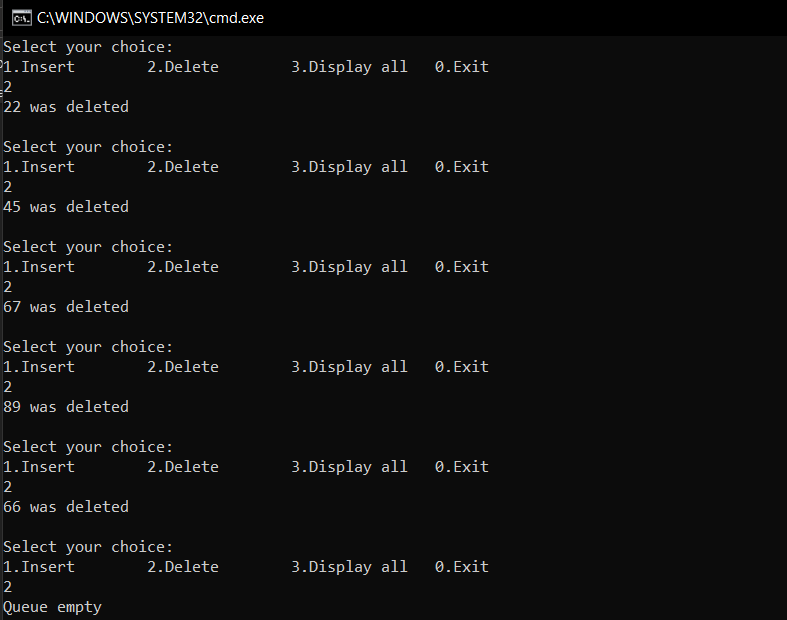
}

Output: Enqueue and display





Output: dequeue till empty



Output: Enqueue and dequeue when front==rear+1 and other cases

