

**Queue using Array**

|  |  |
| --- | --- |
| Name: | Wilson Vidyut Doloy |
| Reg No: | 19BCE1603 |
| Subject: | CSE2003 Data Structures and Algorithms Lab |
| Slot: | L67+L68(Prof. Sandeep Kumar) |
| Date: | 01 August 2020 |

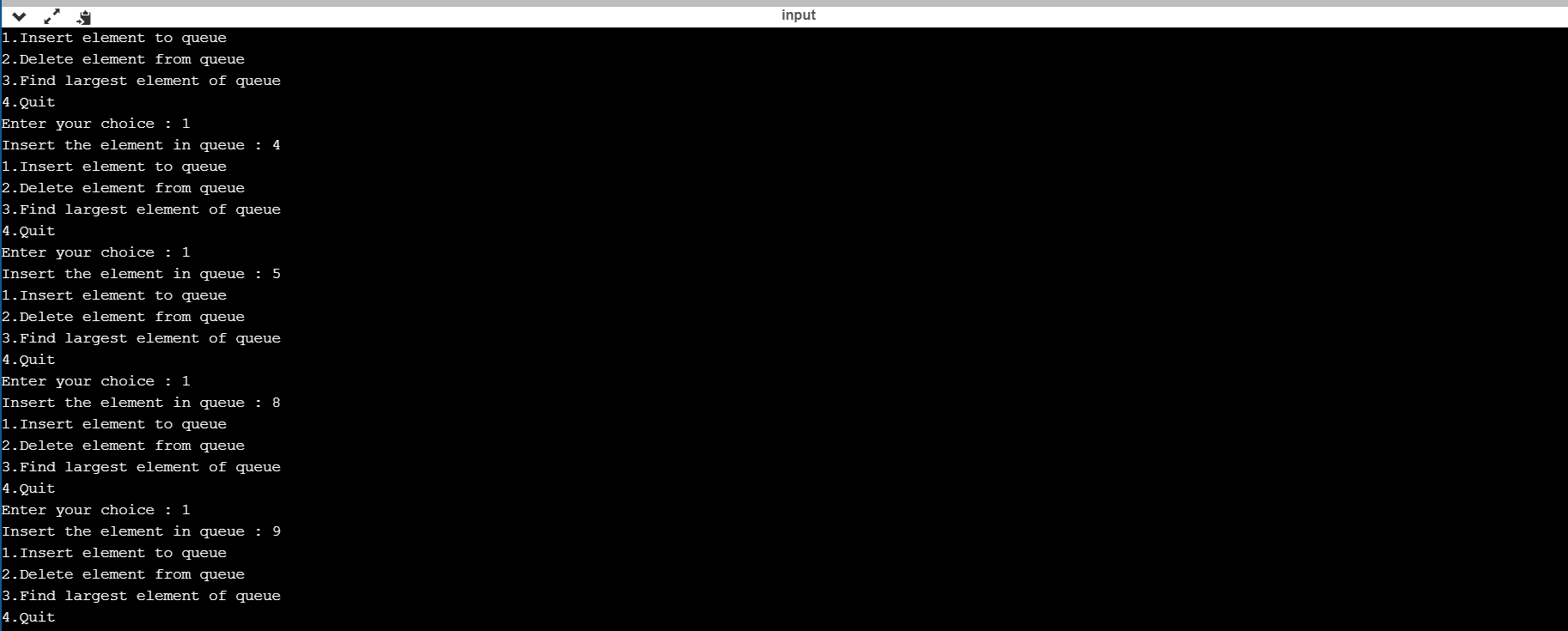
1) Let Q be a queue data structure and it has n numbers. Write C code to implement the following functions on Q

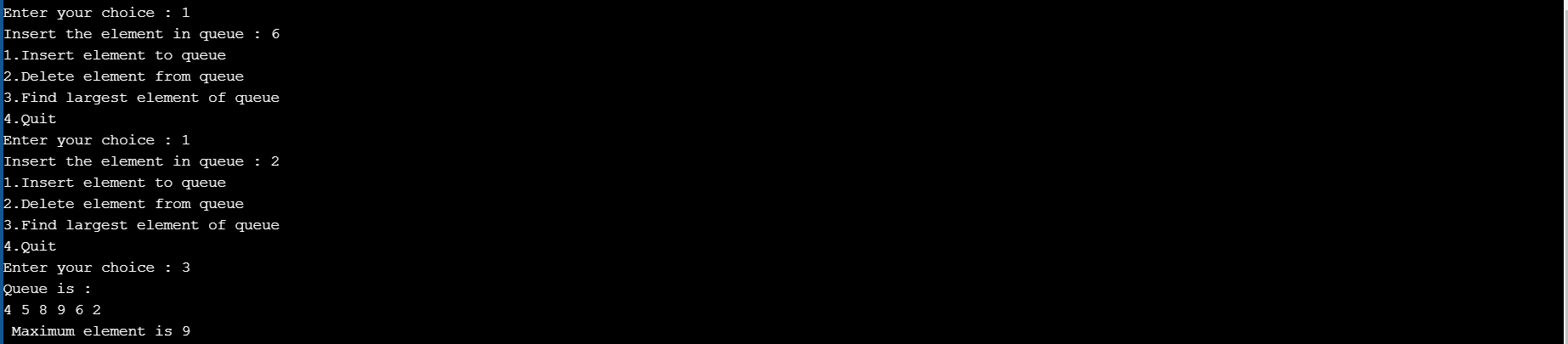
(a) Write a function to compute the maximum number in Q.

CODE:

#include <stdio.h>  
#include<stdlib.h>  
#define MAX 50  
void insert();  
void delete();  
void maximum();  
int queue\_array[MAX];  
int rear = - 1;  
int front = - 1;  
int maxv=0;  
int main()  
{  
int choice;  
while (1)  
{  
printf("1.Insert element to queue \n");  
printf("2.Delete element from queue \n");  
printf("3.Find largest element of queue \n");  
printf("4.Quit \n");  
printf("Enter your choice : ");  
scanf("%d", &choice);  
switch(choice)  
{  
case 1:  
insert();  
break;  
case 2:  
delete();  
break;  
case 3:  
maximum();  
break;  
case 4:  
exit(1);  
default:  
printf("Wrong choice \n");  
}  
}  
}  
void insert()  
{  
int item;  
if(rear == MAX - 1)  
printf("Queue Overflow \n");  
else  
{  
if(front== - 1)  
front = 0;  
printf("Insert the element in queue : ");  
scanf("%d", &item);  
rear = rear + 1;  
if(rear==0)  
{  
 maxv=item;  
}  
if (item>maxv && rear>0)  
{  
 maxv=item;  
}  
queue\_array[rear] = item;  
}  
}  
void delete()  
{  
if(front == - 1 || front > rear)  
{  
printf("Queue Underflow \n");  
return;  
}  
else  
{  
printf("Element deleted from queue is : %d\n", queue\_array[front]);  
front = front + 1;  
}  
}  
void maximum()  
{  
int i;  
if(front == - 1)  
printf("Queue is empty \n");  
else  
{  
printf("Queue is : \n");  
for(i = front; i <= rear; i++)  
printf("%d ", queue\_array[i]);  
printf("\n");  
}  
printf(" Maximum element is %d\n",maxv);  
}

OUTPUT:



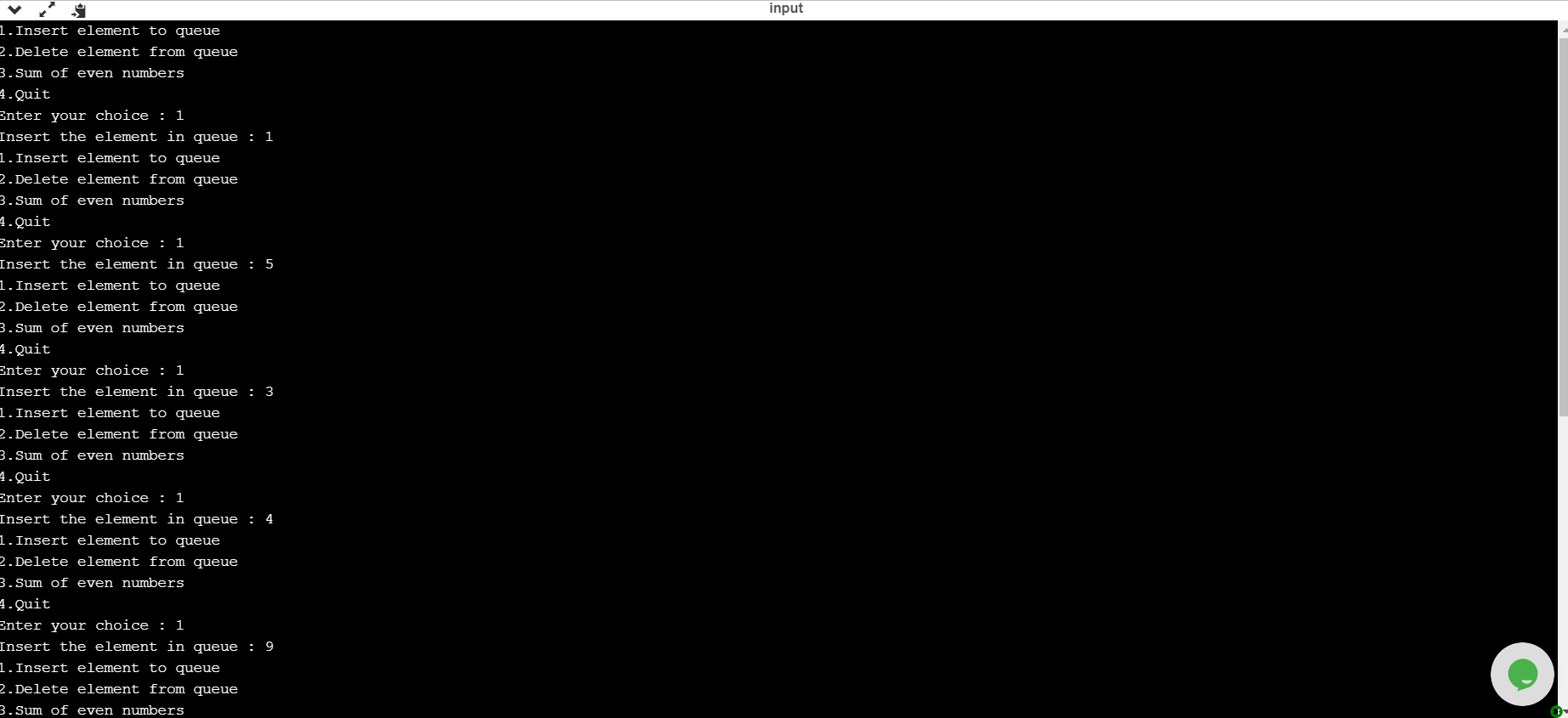


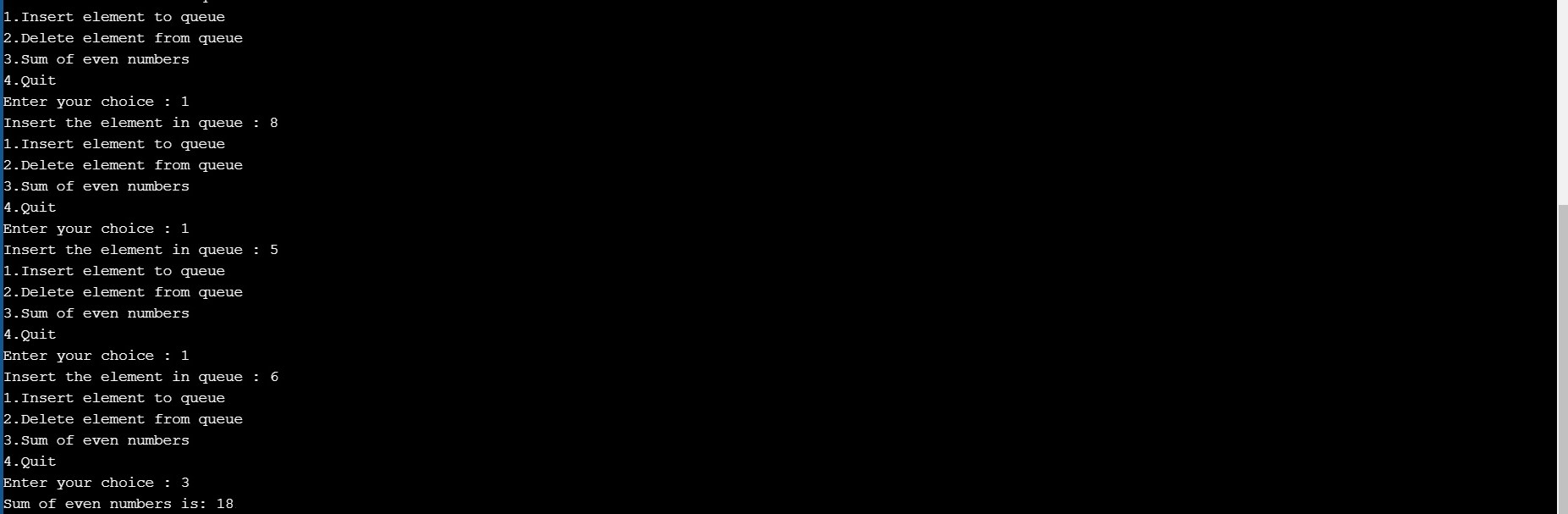
(b) Write a function to compute the sum of the even numbers in Q.

CODE:

void sum\_even()  
{int sum=0;  
 if(front == - 1)  
 printf("Queue is empty \n");  
 else  
 {  
 for(int i=front; i<=rear;i++)  
 {  
 if((queue\_array[i])%2==0)  
 sum+=queue\_array[i];  
 }  
 }  
 printf("Sum of even numbers is: %d",sum);  
   
   
}

OUTPUT:

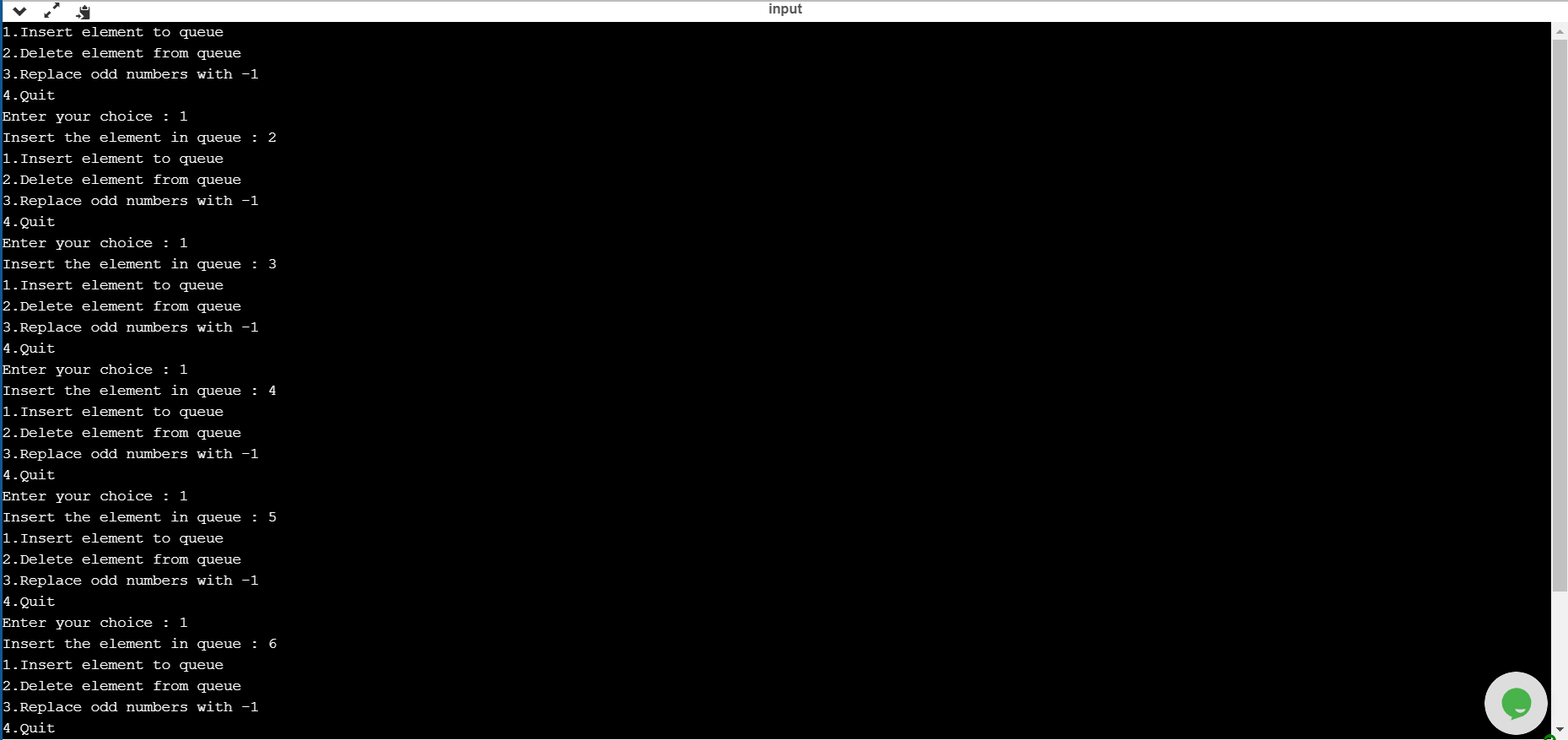


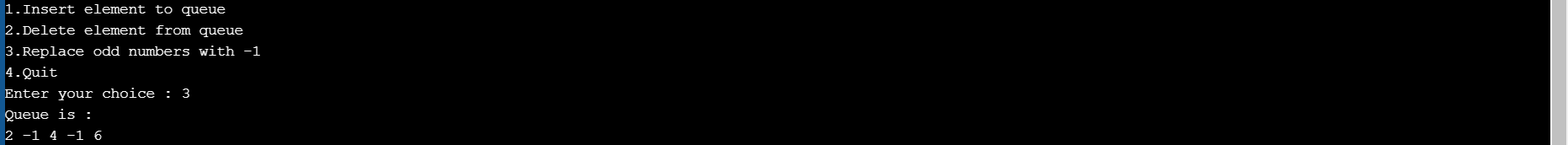


(c) Write a function to replace all odd numbers with -1 in Q.

void replace()  
{  
 if(front == - 1)  
 printf("Queue is empty \n");  
 else  
 {  
 for(int i=front; i<=rear;i++)  
 {  
 if((queue\_array[i])%2!=0)  
 queue\_array[i]=-1;  
 }  
 }  
 printf("Queue is : \n");  
 for(int i = front; i <= rear; i++)  
 printf("%d ", queue\_array[i]);  
   
}

OUTPUT:



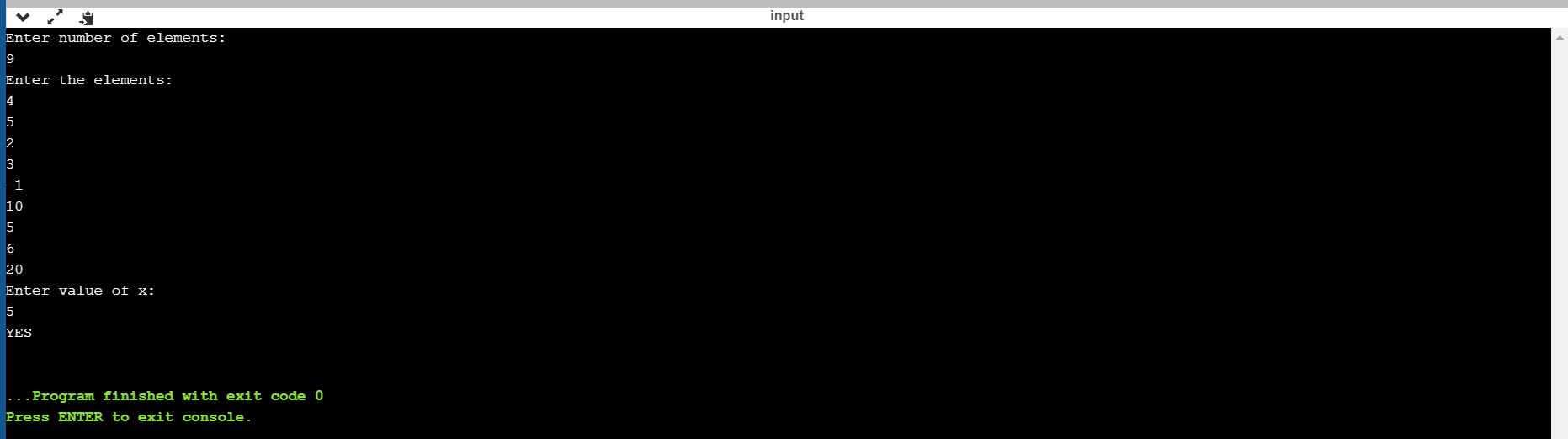


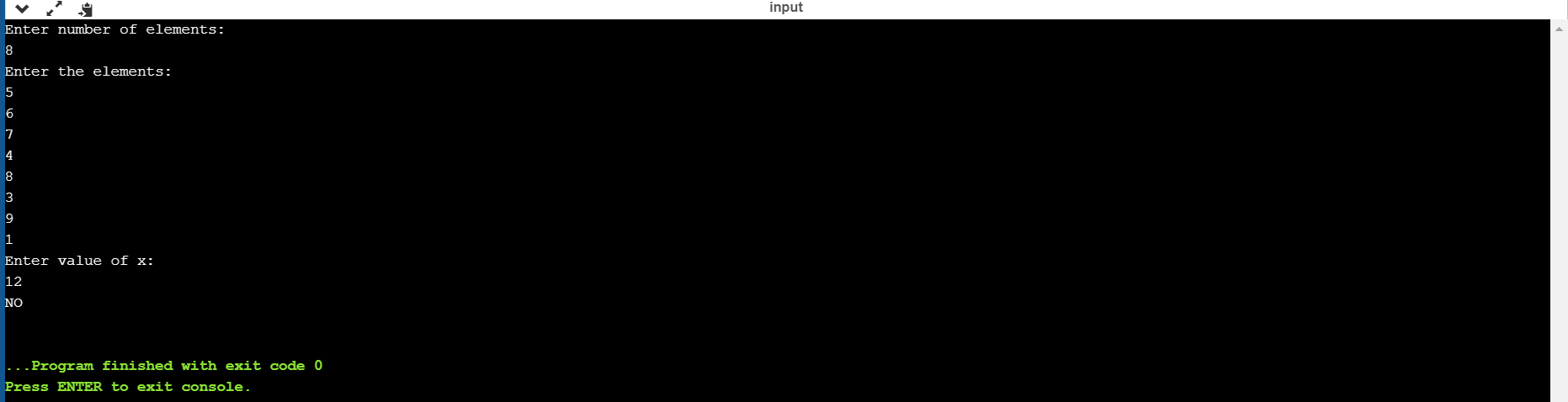
2) Given a stack of integers, using a temporary queue check whether the sum of each successive pair of integers in the stack is greater than or equal to x or not. If the condition satisfies for all pair of numbers in stack then output is true else false. If the stack has an odd number of elements, the element at the top is left out of a pair. For example, if the stack of elements is [4, 5, 2, 3, -1, 10, 5, 6, 20], and x = 5 then the output should be true because sum of each of the pairs (4, 5), (2, 3), (-1, 10), and (5, 6) is greater than 5. And 20 can be discarded.

CODE:

#include <limits.h>  
#include <stdio.h>  
#include <stdlib.h>  
  
struct Stack {  
 int top;  
 unsigned capacity;  
 int\* array;  
};  
   
struct Stack\* createStack(unsigned capacity)  
{  
 struct Stack\* stack = (struct Stack\*)malloc(sizeof(struct Stack));  
 stack->capacity = capacity;  
 stack->top = -1;  
 stack->array = (int\*)malloc(stack->capacity \* sizeof(int));  
 return stack;  
}  
  
int isFulls(struct Stack\* stack)  
{  
 return stack->top == stack->capacity - 1;  
}  
  
int isEmptys(struct Stack\* stack)  
{  
 return stack->top == -1;  
}  
   
void push(struct Stack\* stack, int item)  
{  
 if (isFulls(stack))  
 return;  
 stack->array[++stack->top] = item;  
}  
   
int pop(struct Stack\* stack)  
{  
 if (isEmptys(stack))  
 return INT\_MIN;  
 return stack->array[stack->top--];  
}  
   
int peek(struct Stack\* stack)  
{  
 if (isEmptys(stack))  
 return INT\_MIN;  
 return stack->array[stack->top];  
}  
 struct Queue {  
 int front, rear, size;  
 unsigned capacity;  
 int\* array;  
};  
  
struct Queue\* createQueue(unsigned capacity)  
{  
 struct Queue\* queue = (struct Queue\*)malloc(  
 sizeof(struct Queue));  
 queue->capacity = capacity;  
 queue->front = queue->size = 0;  
   
 queue->rear = capacity - 1;  
 queue->array = (int\*)malloc(  
 queue->capacity \* sizeof(int));  
 return queue;  
}  
   
int isFullq(struct Queue\* queue)  
{  
 return (queue->size == queue->capacity);  
}  
   
int isEmptyq(struct Queue\* queue)  
{  
 return (queue->size == 0);  
}  
   
void enqueue(struct Queue\* queue, int item)  
{  
 if (isFullq(queue))  
 return;  
 queue->rear = (queue->rear + 1)  
 % queue->capacity;  
 queue->array[queue->rear] = item;  
 queue->size = queue->size + 1;  
}  
   
int dequeue(struct Queue\* queue)  
{  
 if (isEmptyq(queue))  
 return INT\_MIN;  
 int item = queue->array[queue->front];  
 queue->front = (queue->front + 1)  
 % queue->capacity;  
 queue->size = queue->size - 1;  
 return item;  
}  
   
int front(struct Queue\* queue)  
{  
 if (isEmptyq(queue))  
 return INT\_MIN;  
 return queue->array[queue->front];  
}  
   
int rear(struct Queue\* queue)  
{  
 if (isEmptyq(queue))  
 return INT\_MIN;  
 return queue->array[queue->rear];  
}  
   
int main()  
{  
 struct Stack\* stack = createStack(1000);  
 struct Queue\* queue = createQueue(1000);  
   
   
 int n,temp,x;  
 printf("Enter number of elements:\n");  
 scanf("%d",&n);  
 printf("Enter the elements:\n");  
 for(int i=0;i<n;i++)  
 {  
 scanf("%d",&temp);  
 push(stack,temp);  
 }  
 printf("Enter value of x:\n");  
 scanf("%d",&x);  
   
   
 while(!isEmptys(stack))  
 {  
 enqueue(queue,pop(stack));  
 }  
   
 //Using queue to reverse order of elements in stack  
 while(!isEmptyq(queue))  
 {  
 push(stack,dequeue(queue));  
 }  
   
 int sum=0,flag=0;  
 while(!isEmptys(stack))  
 {  
 sum+=pop(stack);  
 if(isEmptys(stack))  
 break;  
 sum+=pop(stack);  
 if(sum<x)  
 {flag=1; break;}  
 else  
 sum=0;  
 }  
 if(flag==1)  
 printf("NO\n");  
 else  
 printf("YES\n");  
 return 0;  
}

OUTPUT:





3) Given an integer ‘k’ and a queue of integers, write a program to change the position of the last k elements of the queue by bringing the ‘k’ elements of the queue to the beginning. For example, if k=4 and queue has the elements are [10, 20, 30, 40, 50, 60, 70, 80, 90]; the output should be [60, 70, 80, 90, 10, 20, 30, 40, 50].

CODE:

#include <stdio.h>  
#include<stdlib.h>  
#define MAX 30  
void insert();  
void delete();  
void shift();  
int queue\_array[MAX];  
int rear = - 1;  
int front = - 1;  
int maxv=0;  
int main()  
{  
int choice;  
while (1)  
{  
printf("1.Insert element to queue \n");  
printf("2.Delete element from queue \n");  
printf("3.Shift last k elements to the beginning \n");  
printf("4.Quit \n");  
printf("Enter your choice : ");  
scanf("%d", &choice);  
switch(choice)  
{  
case 1:  
insert();  
break;  
case 2:  
delete();  
break;  
case 3:  
shift();  
break;  
case 4:  
exit(1);  
default:  
printf("Wrong choice \n");  
}  
}  
}  
void insert()  
{  
int item;  
if(rear == MAX - 1)  
printf("Queue Overflow \n");  
else  
{  
if(front== - 1)  
front = 0;  
printf("Insert the element in queue : ");  
scanf("%d", &item);  
rear = rear + 1;  
queue\_array[rear] = item;  
}  
}  
void delete()  
{  
if(front == - 1 || front > rear)  
{  
printf("Queue Underflow \n");  
return;  
}  
else  
{  
printf("Element deleted from queue is : %d\n", queue\_array[front]);  
front = front + 1;  
}  
}  
void shift()  
{  
int i,k,temp;  
printf("Enter values of k: ");  
scanf("%d",&k);  
  
if(front == - 1)  
printf("Queue is empty \n");  
else  
{for(int j=0;j<k;j++)  
{  
temp=queue\_array[rear];  
for(i = rear; i > 0; i--)  
{  
 queue\_array[i]=queue\_array[i-1];  
}  
queue\_array[front]=temp;  
}  
}  
for(i = front; i <= rear; i++)  
{  
 printf("%d ", queue\_array[i]);  
  
}  
printf("\n");  
}

OUTPUT:

