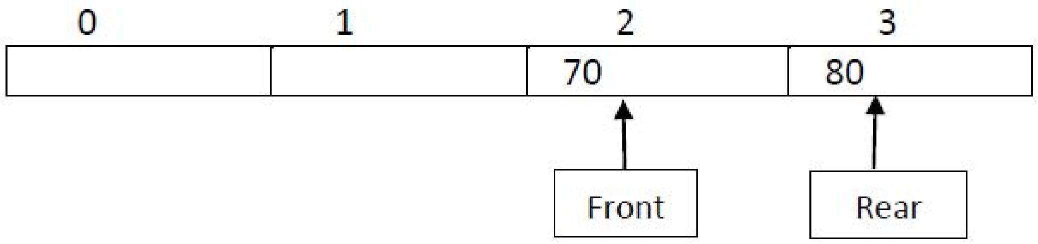
NAME:KUSUM M R **DATE:18/01/2021**
USN:1BM19CS077

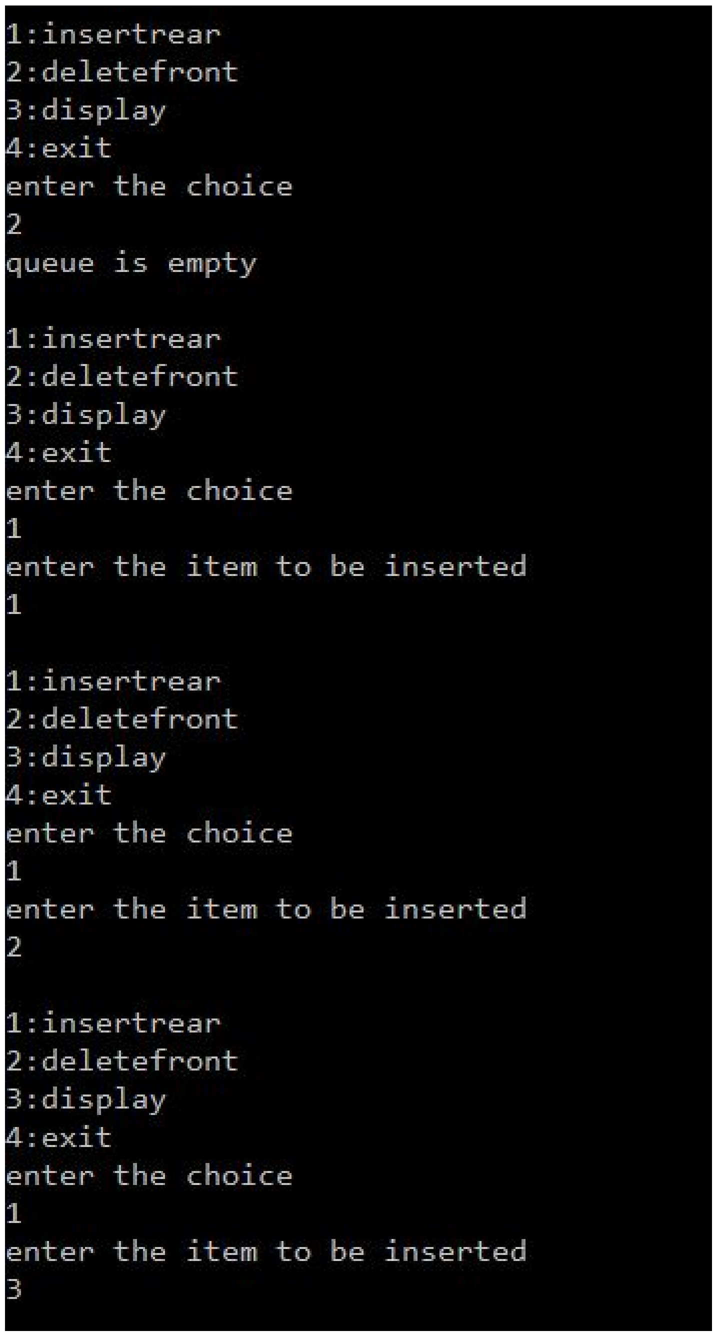
PRACTICE PROGRAM- 6

Consider the below scenario of linear queue. Design and implement a queue which utilize the free space without shifting the elements of queue.

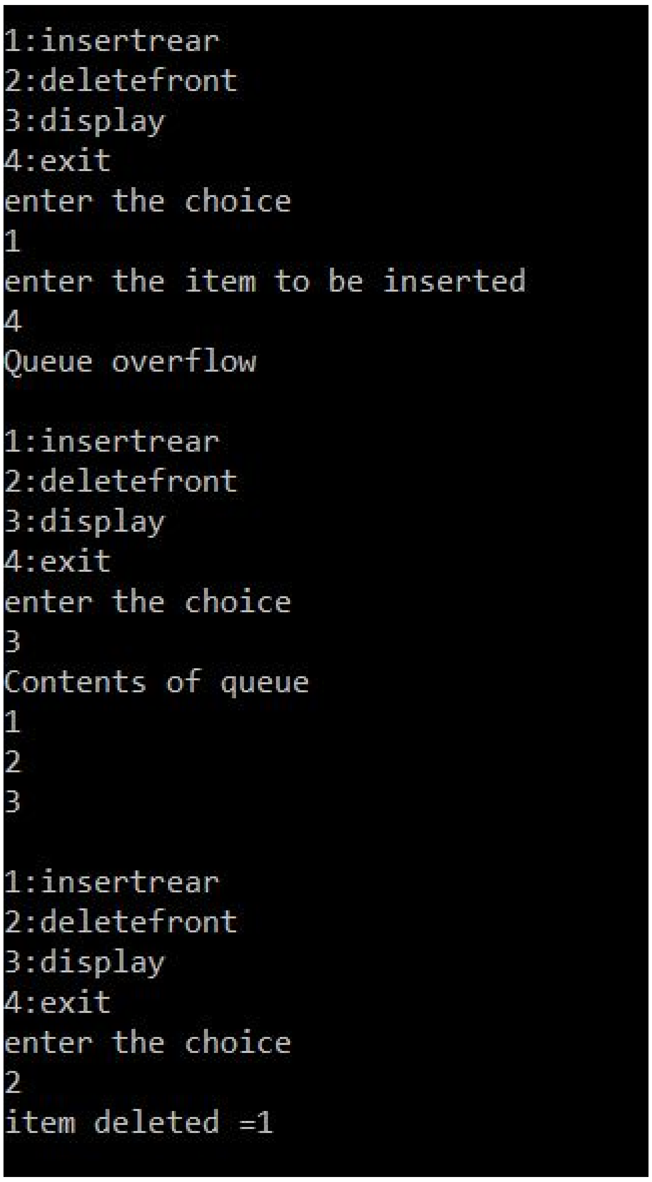


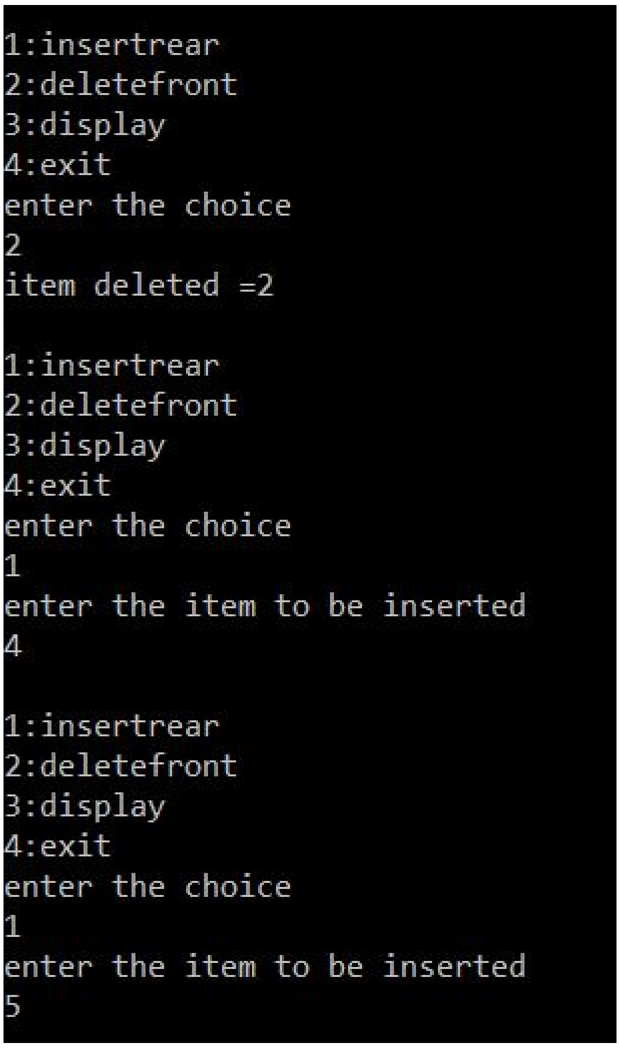
SOURCE CODE:
```
#include<stdio.h>
#include<stdlib.h>
#define QUE_SIZE 3
int item,front=0,rear=-1,q[QUE_SIZE],count=0;
void insertrear()
{
if(count==QUE_SIZE)
{
printf("queue overflow\n");
return;
}
rear=(rear+1)%QUE_SIZE;
q[rear]=item;
count++;
}
int deletefront()
{
if(count==0) return -1;
item=q[front];
front=(front+1)%QUE_SIZE;
count=count-1;
return item;
}
void displayQ()
{
int i,f;
if(count==0)
{
printf("queue is empty\n");
return;
}
f=front;
printf("Contents of queue \n");
for(i=1;i<=count;i++)
{
printf("%d\n",q[f]);
f=(f+1)%QUE_SIZE;
}
}
void main()
{
 int choice;

 for(;;)
 {
 printf("\n1:insertrear\n2:deletefront\n3:display\n4:exit\n");
 printf("enter the choice\n");
 scanf("%d",&choice);

 switch(choice)
 {
 case 1:printf("enter the item to be inserted\n");
      scanf("%d",&item);
      insertrear();
      break;
 case 2:item=deletefront();
      if(item==-1)
      printf("queue is empty\n");
      else
      printf("item deleted =%d\n",item);
      break;
 case 3:displayQ();
      break;
 case 4:exit(0);
 break;
 default:printf("Invalid choice\n");
}
}
}
```

OUTPUT:

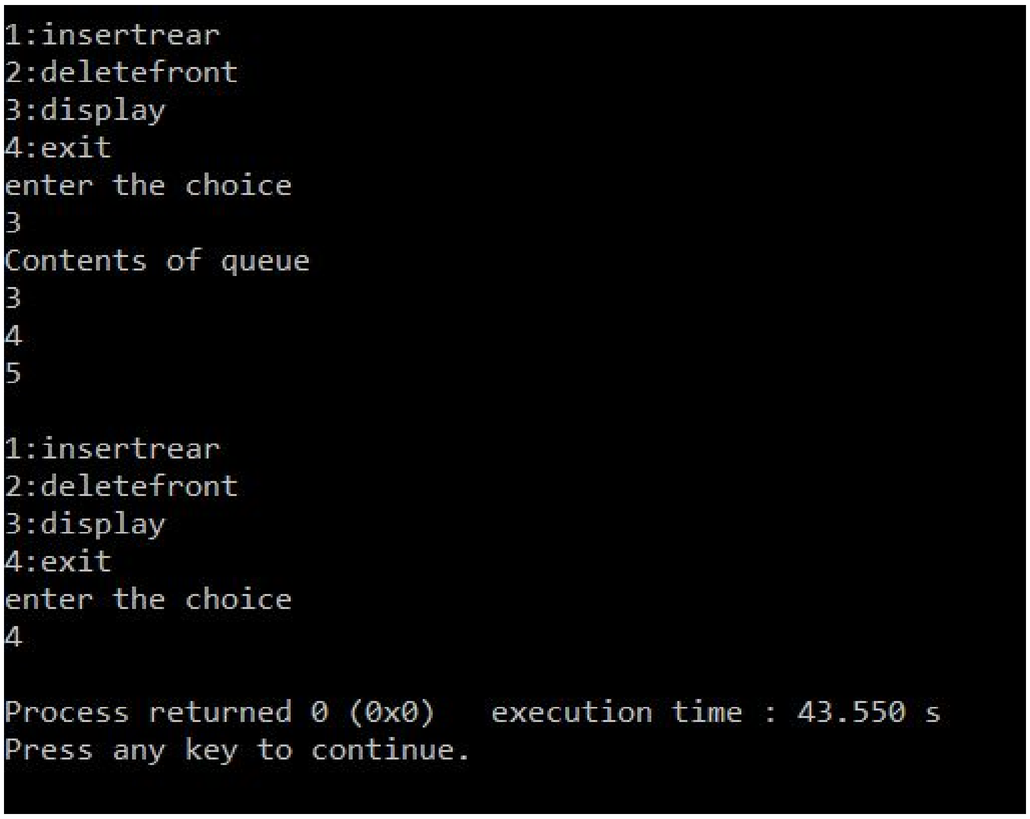


```
1:insertrear
2:deletefront
3:display
4:exit
enter the choice
2
queue is empty

1:insertrear
2:deletefront
3:display
4:exit
enter the choice
1
enter the item to be inserted
1

1:insertrear
2:deletefront
3:display
4:exit
enter the choice
1
enter the item to be inserted
2

1:insertrear
2:deletefront
3:display
4:exit
enter the choice
1
enter the item to be inserted
3
```


```
1:insertrear
2:deletefront
3:display
4:exit
enter the choice
1
enter the item to be inserted
4
Queue overflow

1:insertrear
2:deletefront
3:display
4:exit
enter the choice
3
Contents of queue
1
2
3

1:insertrear
2:deletefront
3:display
4:exit
enter the choice
2
item deleted =1
```



```
1:insertrear
2:deletefront
3:display
4:exit
enter the choice
2
item deleted =2

1:insertrear
2:deletefront
3:display
4:exit
enter the choice
1
enter the item to be inserted
4

1:insertrear
2:deletefront
3:display
4:exit
enter the choice
1
enter the item to be inserted
5
```



```
1:insertrear
2:deletefront
3:display
4:exit
enter the choice
3
Contents of queue
3
4
5

1:insertrear
2:deletefront
3:display
4:exit
enter the choice
4

Process returned 0 (0x0)   execution time : 43.550 s
Press any key to continue.
```

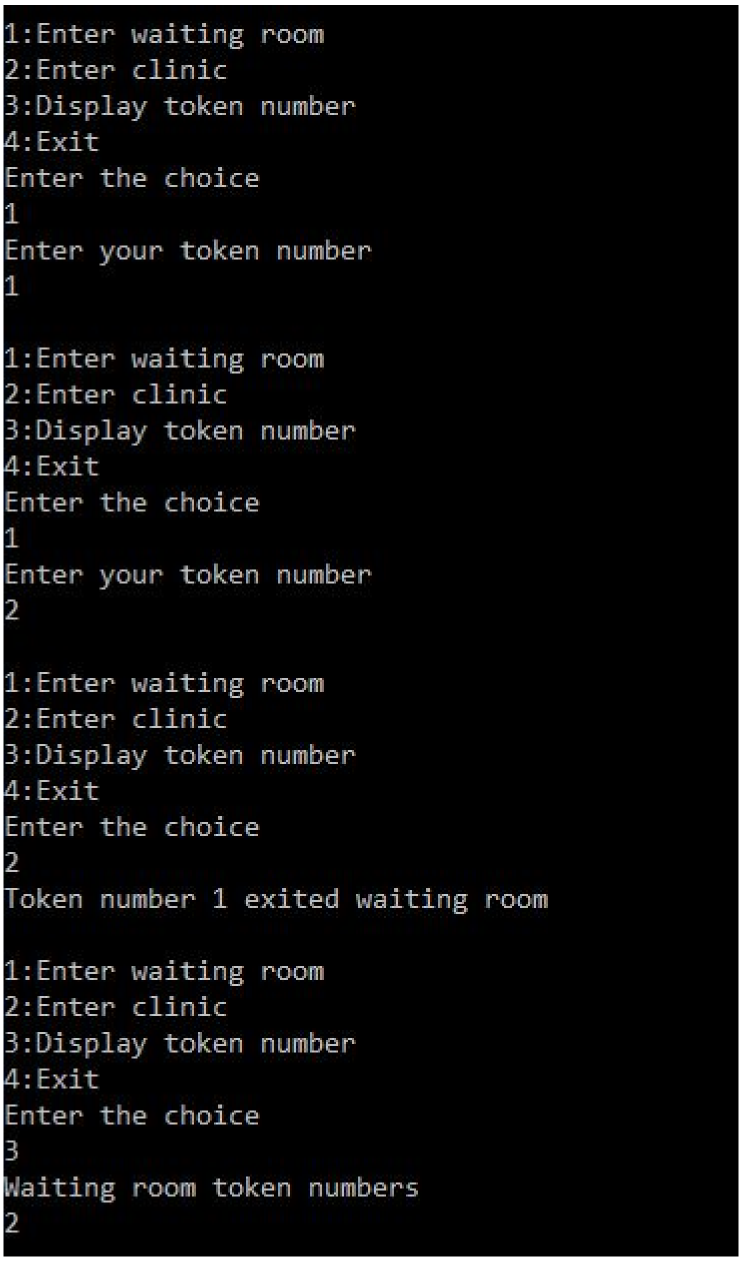
PRACTICE PROGRAM- 7

The XYZ clinic has a waiting room with ten chairs. The chairs available in the clinic should be designed to occupy based on the FIFO architecture and also more efficiently by allowing customers to occupy the seats vacantly. In the case of no vacant seats in the waiting room, a "Waiting Room Full" message should be displayed at the reception counter. Develop an application(C Program) for the above scenario with a suitable data structure.

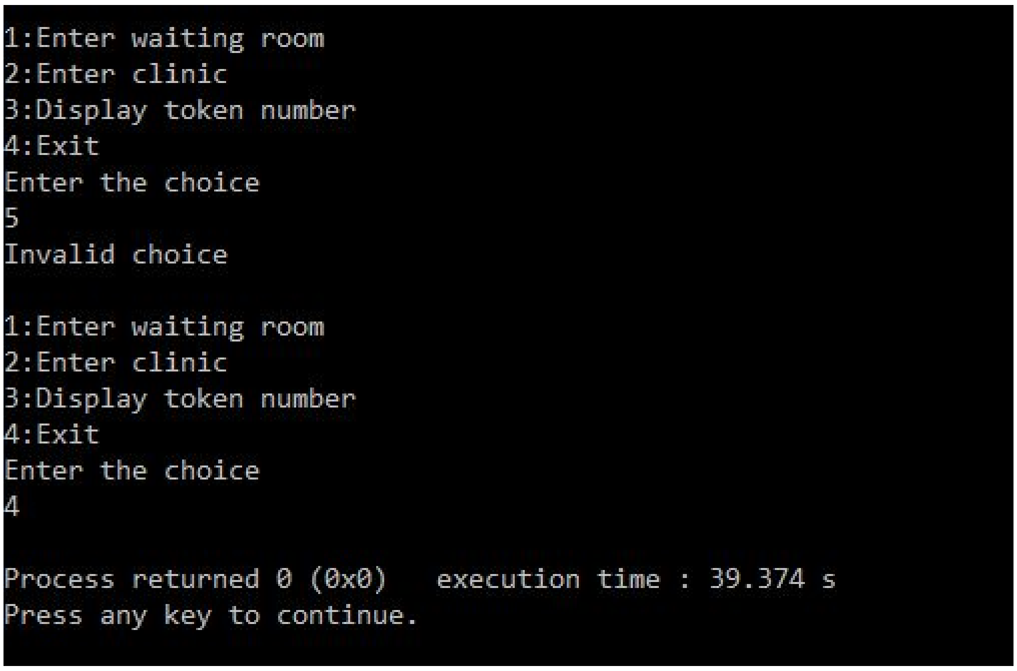
SOURCE CODE:

```
#include<stdio.h>
#include<stdlib.h>
#define QUE_SIZE 3
int item,front=0,rear=-1,q[QUE_SIZE],count=0;
void insertrear()
{
if(count==QUE_SIZE)
{
printf("Waiting Room Full\n");
return;
}
rear=(rear+1)%QUE_SIZE;
q[rear]=item;
count++;
}
int deletefront()
{
if(count==0) return -1;
item=q[front];
front=(front+1)%QUE_SIZE;
count=count-1;
return item;
}
void displayQ()
{
int i,f;
if(count==0)
{
printf("Vacant waiting room\n");
return;
}
f=front;
printf("Waiting room token numbers \n");
for(i=1;i<=count;i++)
{
printf("%d\n",q[f]);
f=(f+1)%QUE_SIZE;
}
}
void main()
{
 int choice;

 for(;;)
 {
 printf("\n1:Enter waiting room\n2:Enter clinic\n3:Display token number\n4:Exit\n");
 printf("Enter the choice\n");
 scanf("%d",&choice);

 switch(choice)
 {
 case 1:printf("Enter your token number\n");
      scanf("%d",&item);
      insertrear();
      break;
 case 2:item=deletefront();
      if(item==-1)
      printf("Waiting Room Full\n");
      else
      printf("Token number %d exited waiting room\n",item);
      break;
 case 3:displayQ();
      break;
 case 4:exit(0);
 break;
 default:printf("Invalid choice\n");
}
}
}
```

OUTPUT:



```
1:Enter waiting room
2:Enter clinic
3:Display token number
4:Exit
Enter the choice
1
Enter your token number
1

1:Enter waiting room
2:Enter clinic
3:Display token number
4:Exit
Enter the choice
1
Enter your token number
2

1:Enter waiting room
2:Enter clinic
3:Display token number
4:Exit
Enter the choice
2
Token number 1 exited waiting room

1:Enter waiting room
2:Enter clinic
3:Display token number
4:Exit
Enter the choice
3
Waiting room token numbers
2
```



```
1:Enter waiting room
2:Enter clinic
3:Display token number
4:Exit
Enter the choice
5
Invalid choice

1:Enter waiting room
2:Enter clinic
3:Display token number
4:Exit
Enter the choice
4

Process returned 0 (0x0)   execution time : 39.374 s
Press any key to continue.
```

PRACTICE PROGRAM- 8

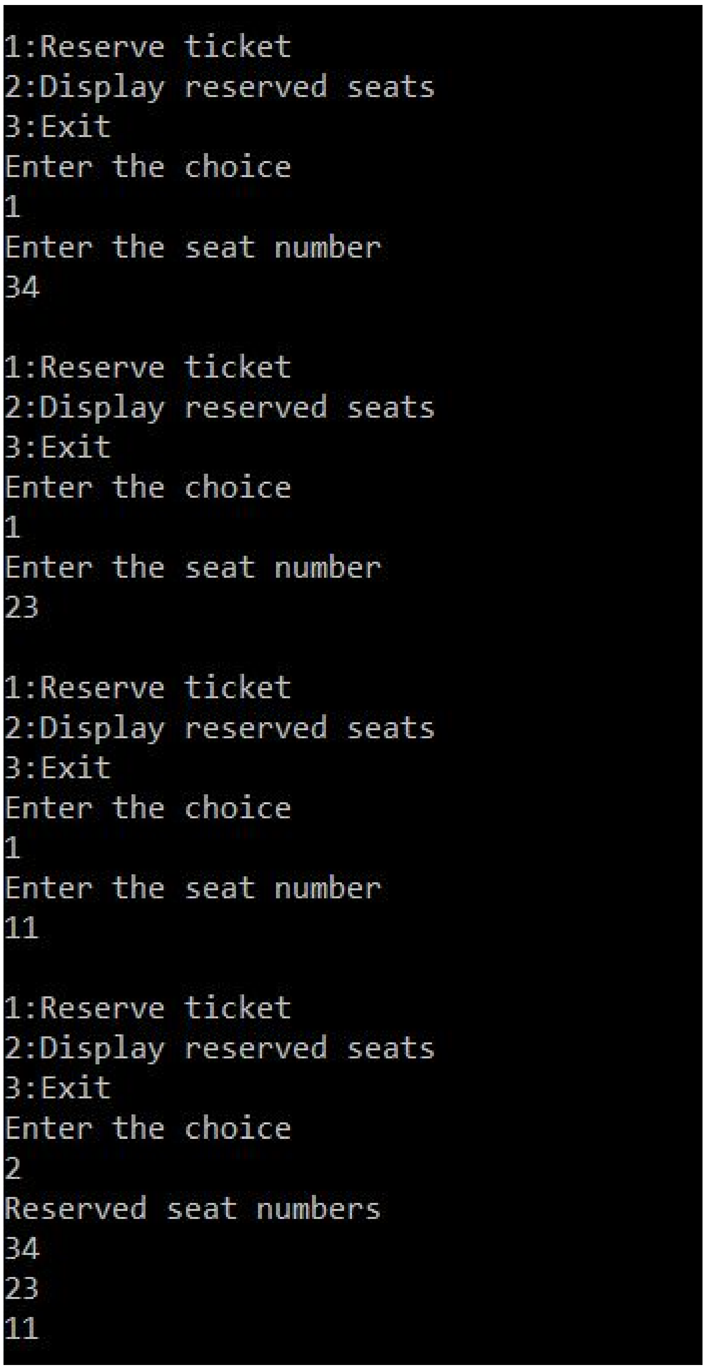
Develop an application (C Program) with suitable data structure to demonstrate the Online Movie Ticket Reservation system, in which users request should get process on the basis of First come First basis and display “Reservation Full”, “Reservation Started” appropriately based on the availability of the Tickets.

SOURCE CODE:

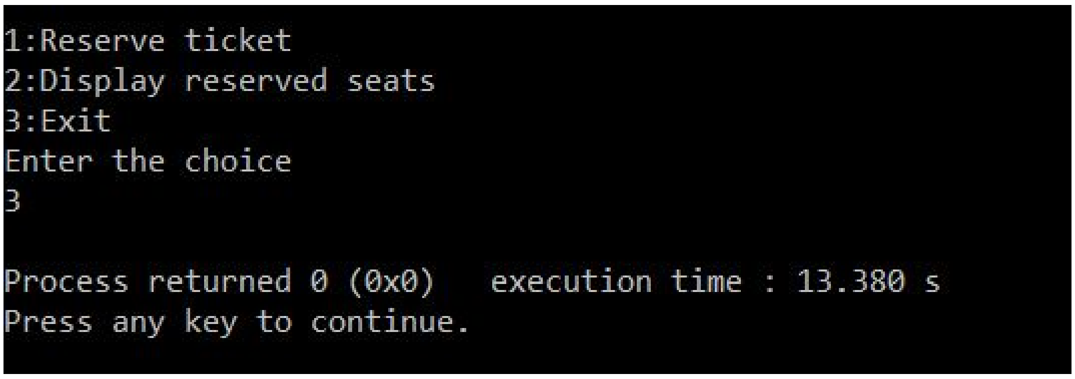
```
#include<stdio.h>
#include<process.h>
#define QUE_SIZE 3
int item,front=0,rear=-1,q[10];
void insertrear()
{

   if(rear==QUE_SIZE-1)
   {
      printf("Reservation Full\n");
      return ;
   }
   rear=rear+1;
   q[rear]=item;
}
void displayQ()
{
   int i;
   if(front>rear)
   {
      printf("Reservation Started\n");
      return ;
   }
```

```
    printf("Reserved seat numbers\n");
    for(i=front;i<=rear;i++)
    {
        printf("%d\n",q[i]);
    }
}
void main()
{
    int choice;
    for(;;)
    {
        printf("\n1:Reserve ticket\n2:Display reserved seats\n3:Exit\n");
        printf("Enter the choice\n");
        scanf("%d",&choice);
        switch(choice)
        {
            case 1:printf("Enter the seat number\n");
            scanf("%d",&item);
            insertrear();
            break;
            case 2:displayQ();
            break;
            default:exit(0);
        }
    }
}
```

OUTPUT:



```
1:Reserve ticket
2:Display reserved seats
3:Exit
Enter the choice
1
Enter the seat number
34

1:Reserve ticket
2:Display reserved seats
3:Exit
Enter the choice
1
Enter the seat number
23

1:Reserve ticket
2:Display reserved seats
3:Exit
Enter the choice
1
Enter the seat number
11

1:Reserve ticket
2:Display reserved seats
3:Exit
Enter the choice
2
Reserved seat numbers
34
23
11
```



```
1:Reserve ticket
2:Display reserved seats
3:Exit
Enter the choice
3

Process returned 0 (0x0)   execution time : 13.380 s
Press any key to continue.
```

PRACTICE PROGRAM- 9

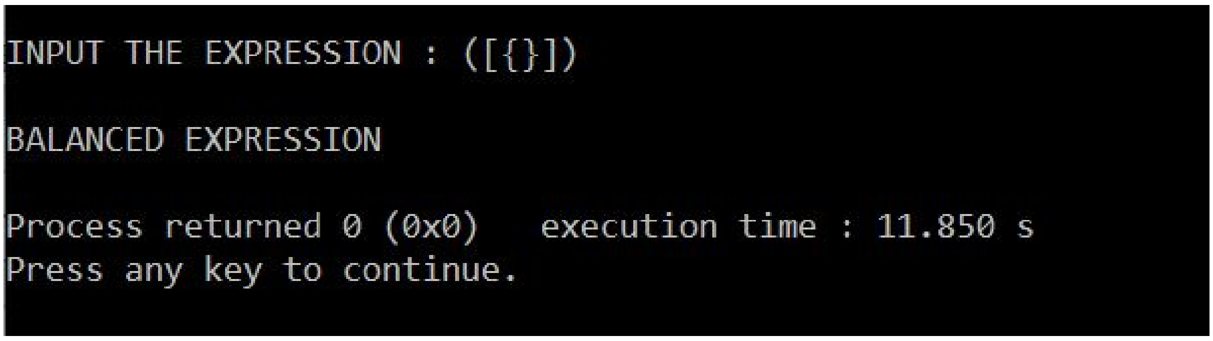
Design and implement the below given problem statement with most suitable data structure. The balanced bracket problem is to recognize sentences composed of sequences of two symbols, (and), which are correctly nested. E.g. (()) is correctly nested but ())is not. A limit of ten symbols per sentence was assumed.

SOURCE CODE:

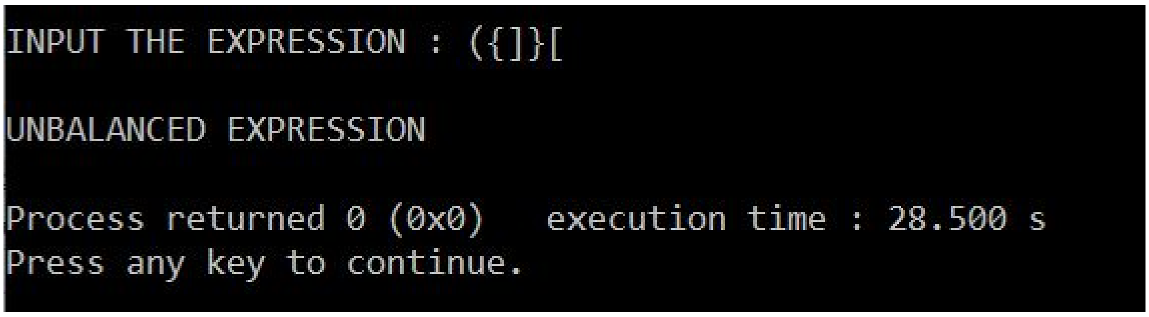
```
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
#define MAX 10

struct stack
{
char stk[MAX];
int top;
}s;

void push(char item)
{
if (s.top == (MAX - 1))
printf ("Stack is Full\n");
else
{
s.top = s.top + 1;
s.stk[s.top] = item;
}}

void pop()
{
if (s.top == - 1)
{
printf ("Stack is Empty\n");
}
else
{
s.top = s.top - 1; // Pop the char and decrement top
}}

int main()
{
char exp[MAX];
int i = 0;
s.top = -1;
printf("\nINPUT THE EXPRESSION : ");
scanf("%s", exp);
for(i = 0;i < strlen(exp);i++)
{
```

```
if(exp[i] == '(' || exp[i] == '[' || exp[i] == '{')
{
push(exp[i]);
continue;
}
else if(exp[i] == ')' || exp[i] == ']' || exp[i] == '}')
{
if(exp[i] == ')')
{
if(s.stk[s.top] == '(')
{
pop();
}
else
{
printf("\nUNBALANCED EXPRESSION\n");
break;
}}
if(exp[i] == ']')
{
if(s.stk[s.top] == '[')
{
pop();
}
else
{
printf("\nUNBALANCED EXPRESSION\n");
break;
}}
if(exp[i] == '}')
{
if(s.stk[s.top] == '{')
{
pop();
}
else
{
printf("\nUNBALANCED EXPRESSION\n");
break;
}}}}
if(s.top == -1)
{
printf("\nBALANCED EXPRESSION\n");
}}
```

OUTPUT:



```
INPUT THE EXPRESSION : ([{}])

BALANCED EXPRESSION

Process returned 0 (0x0)   execution time : 11.850 s
Press any key to continue.
```



```
INPUT THE EXPRESSION : ({]}[

UNBALANCED EXPRESSION

Process returned 0 (0x0)   execution time : 28.500 s
Press any key to continue.
```