

Stack and Queue

Swaminathan Navinsashok
2019115126

Stack

SOURCE CODE

```
#include<stdio.h>
#include<stdlib.h>
#define EMPTY_TOS (-1)
#define MIN_STACK_SIZE 2

struct stack_record
{
    int capacity;
    int tos;
    int *starr;    // element_type starr[100] ;
};

typedef struct stack_record *stack;

stack create_stack(int max_elements )
{
    stack s;

    if( max_elements < MIN_STACK_SIZE )
        printf("\n Stack size is too small \n ");

    s = (struct stack_record *) malloc( sizeof( struct stack_record ) );

    if( s == NULL )
        printf(" \n Out of space!!! \n ");

    s->starr = (int *)malloc( sizeof( int ) * max_elements );

    if( s->starr == NULL )
```

```

        printf("\n Out of space!!! \n ");

        s->tos = EMPTY_TOS;
        s->capacity = max_elements;
        return( s );
    }
    // Free the memory
void dispose(stack s)
{
    if(s == NULL)
        printf("\n stack is not available \n ");
    else
    {
        free(s->starr);
        free(s);
    }
}

int isEmpty(stack s)
{
    return ( s->tos == EMPTY_TOS );
}

int isFull(stack s)
{
    return( s->tos == s->capacity -1 );
}

//pushing a new element
void push(stack s, int x)
{
    if( !isFull(s))
        s->starr[++s->tos] = x;    // ++s->tos;    s->starr[s->tos] = x ;
    else
        printf("\n overflow \n");
}

void pop(stack s)
{
    if(isEmpty(s))
        printf("\n stack underflow \n ");
    else
        s->tos-- ;
}

```

```

void top(stack s)
{
    if(!isEmpty(s))
        printf("\n top element %d \n ", s->starr[s->tos] );
    else
        printf("\n stack is empty \n");
}

int tap( stack s)
{
    if(!isEmpty(s))
    {
        top(s);
        pop(s);
    }
    else
    {
        printf("\n empty and underflow \n");
    }
}

int main()
{
    int c;
    stack s;
    printf("\n enter 0 to exit\n 1 to create stack \n 2 to dispose stack \n 3 to print top \n 4 to push \n 5 to pop\n 6 top and pop \n");
    scanf("%d",&c);
    do
    {
        switch(c)
        {
            case 0:{
                printf("\n exiting\n");
                break;
            }

            case 1:{
                int me;printf("\n enter no. of elements \n");
                scanf("%d",&me);
                s=create_stack(me);
            }
        }
    } while(c!=0);
}

```

```

        break;
    }
    case 2:{
        dispose(s);
        break;
    }
    case 3:{
        top(s);

        break;
    }
    case 4:{
        int n;
        printf("\n enter element to push \n");
        scanf("%d",&n);
        push(s,n);
        break;
    }
    case 5:{
        pop(s);

        break;
    }
    case 6:{
        tap(s);

        break;
    }
    default: {
        printf("\n only 0 to 5\n");
        break;}

}
if(c==0) break;

```

```

        printf("\n enter 0 to exit\n 1 to create stack \n 2 to dispose stack
\n 3 to print top \n 4 to push \n 5 to pop\n 6 top and pop \n");
        scanf("%d",&c);

```

```
    }while(c!=0);  
    return 0;  
  
}
```

SCREENSHOT OF CODE

Visual Studio Code interface showing a C program for a stack implementation. The Explorer sidebar on the left shows the file structure with folders for 'masum.c', 'msum.c', 'msump.c', 'sp.c', 'str.c', 'st.c', 'lq.c', 'stk.c', and 'scopy.c'. The main editor displays the code for 'stk.c' in the 'main()' function, which uses a switch statement to handle various stack operations. The code is as follows:

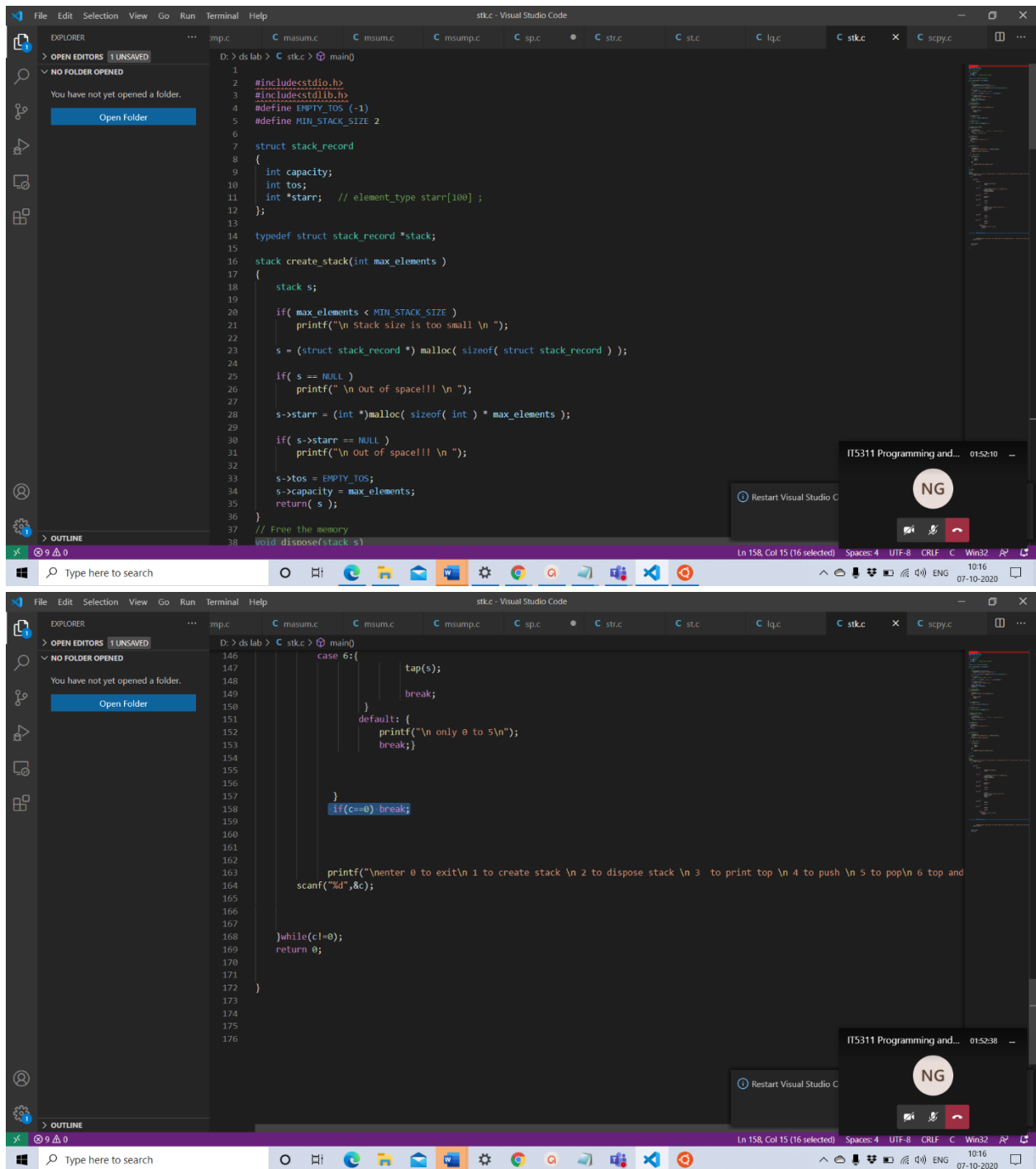
```
126     }
127     case 3:{
128         top(s);
129         break;
130     }
131     case 4:{
132         int n;
133         printf("\n enter element to push \n");
134         scanf("%d",&n);
135         push(s,n);
136         break;
137     }
138     case 5:{
139         pop(s);
140         break;
141     }
142     case 6:{
143         tap(s);
144         break;
145     }
146     default: {
147         printf("\n only 0 to 5\n");
148         break;
149     }
150 }
151 if(c==0) break;
152 printf("\n enter 0 to exit\n 1 to create stack \n 2 to dispose stack \n 3 to print top \n 4 to push \n 5 to pop \n 6 to top and pop \n");
153 scanf("%d",&c);
154 do
155 {
156     switch(c)
157     {
158         case 0:{
159             printf("\nexiting\n");
160             break;
161         }
162         case 1:{
163             int me;printf("\n enter no. of elements \n");
164             scanf("%d",&me);
165             s=create_stack(me);
166             break;
167         }
168         case 2:{
169             dispose(s);
170             break;
171         }
172         case 3:{
173             top(s);
174             break;
175         }
176         case 4:{
177             int n;
178             printf("\n enter element to push \n");
179             scanf("%d",&n);
180             push(s,n);
181             break;
182         }
183         case 5:{
184             pop(s);
185             break;
186         }
187         case 6:{
188             tap(s);
189             break;
190         }
191         default: {
192             printf("\n only 0 to 5\n");
193             break;
194         }
195     }
196     if(c==0) break;
197     printf("\n enter 0 to exit\n 1 to create stack \n 2 to dispose stack \n 3 to print top \n 4 to push \n 5 to pop \n 6 to top and pop \n");
198     scanf("%d",&c);
199 }
200 while(c!=0);
201 }
```

Visual Studio Code interface showing the same C program for a stack implementation. The Explorer sidebar on the left shows the file structure with folders for 'masum.c', 'msum.c', 'msump.c', 'sp.c', 'str.c', 'st.c', 'lq.c', 'stk.c', and 'scopy.c'. The main editor displays the code for 'stk.c' in the 'main()' function, which uses a switch statement to handle various stack operations. The code is as follows:

```
99 int main()
100 {
101     int c;
102     stack s;
103     printf("\n enter 0 to exit\n 1 to create stack \n 2 to dispose stack \n 3 to print top \n 4 to push \n 5 to pop \n 6 to top and pop \n");
104     scanf("%d",&c);
105     do
106     {
107         switch(c)
108         {
109             case 0:{
110                 printf("\nexiting\n");
111                 break;
112             }
113             case 1:{
114                 int me;printf("\n enter no. of elements \n");
115                 scanf("%d",&me);
116                 s=create_stack(me);
117                 break;
118             }
119             case 2:{
120                 dispose(s);
121                 break;
122             }
123             case 3:{
124                 top(s);
125                 break;
126             }
127             case 4:{
128                 int n;
129                 printf("\n enter element to push \n");
130                 scanf("%d",&n);
131                 push(s,n);
132                 break;
133             }
134             case 5:{
135                 pop(s);
136                 break;
137             }
138             case 6:{
139                 tap(s);
140                 break;
141             }
142             default: {
143                 printf("\n only 0 to 5\n");
144                 break;
145             }
146         }
147         if(c==0) break;
148         printf("\n enter 0 to exit\n 1 to create stack \n 2 to dispose stack \n 3 to print top \n 4 to push \n 5 to pop \n 6 to top and pop \n");
149         scanf("%d",&c);
150     }
151     while(c!=0);
152 }
```

```
67
68 void pop(stack s)
69 {
70     if(!isEmpty(s))
71         printf("\\n stack underflow \\n ");
72     else
73         s->tos--;
74 }
75
76
77 void top(stack s)
78 {
79     if(!isEmpty(s))
80         printf("\\n top element %d \\n ", s->starr[s->tos]);
81     else
82         printf("\\n stack is empty \\n");
83 }
84
85 int tap( stack s)
86 {
87     if(!isEmpty(s))
88     {
89         top(s);
90         pop(s);
91     }
92     else
93     {
94         printf("\\n empty and underflow \\n");
95     }
96 }
97
98
99 int main()
100 {
101     int c;
102     stack s;
103     printf("\\n enter 0 to exit\\n 1 to create stack \\n 2 to dispose stack \\n 3 to print top \\n 4
104         scanf(\"%d\",&c);
```

```
37 // Free the memory
38 void dispose(stack s)
39 {
40     if(s == NULL)
41         printf("\\n stack is not available \\n ");
42     else
43     {
44         free(s->starr);
45         free(s);
46     }
47 }
48
49 int isEmpty(stack s)
50 {
51     return ( s->tos == EMPTY_TOS );
52 }
53
54 int isFull(stack s)
55 {
56     return( s->tos == s->capacity -1 );
57 }
58
59 //pushing a new element
60 void push(stack s, int x)
61 {
62     if( !isFull(s) )
63         s->starr[++s->tos] = x; // ++s->tos; s->starr[s->tos] = x ;
64     else
65         printf("\\n overflow \\n");
66 }
67
68 void pop(stack s)
69 {
70     if(!isEmpty(s))
71         printf("\\n stack underflow \\n ");
72     else
73         s->tos--;
```



Output screenshot


```
navin@DESKTOP-D400C38: /mnt/d/ds lab
enter element to push
3

enter 0 to exit
1 to create stack
2 to dispose stack
3 to print top
4 to push
5 to pop
6 top and pop
4

enter element to push
4

enter 0 to exit
1 to create stack
2 to dispose stack
3 to print top
4 to push
5 to pop
6 top and pop
4

enter element to push
5

enter 0 to exit
1 to create stack
2 to dispose stack
3 to print top
4 to push
5 to pop
6 top and pop
4

enter element to push
6

overflow

enter 0 to exit
1 to create stack
2 to dispose stack
3 to print top
4 to push
5 to pop
6 top and pop
6
```

```
navin@DESKTOP-D400C38: /mnt/d/ds lab
6 top and pop
6

top element 5

enter 0 to exit
1 to create stack
2 to dispose stack
3 to print top
4 to push
5 to pop
6 top and pop
6

top element 4

enter 0 to exit
1 to create stack
2 to dispose stack
3 to print top
4 to push
5 to pop
6 top and pop
6

top element 3

enter 0 to exit
1 to create stack
2 to dispose stack
3 to print top
4 to push
5 to pop
6 top and pop
6

top element 2

enter 0 to exit
1 to create stack
2 to dispose stack
3 to print top
4 to push
5 to pop
6 top and pop
5

enter 0 to exit
1 to create stack
2 to dispose stack
```

```
navin@DESKTOP-D400C38: /mnt/d/ds lab
5 to pop
6 top and pop
5

enter 0 to exit
1 to create stack
2 to dispose stack
3 to print top
4 to push
5 to pop
6 top and pop
3

stack is empty

enter 0 to exit
1 to create stack
2 to dispose stack
3 to print top
4 to push
5 to pop
6 top and pop
6

empty and underflow

enter 0 to exit
1 to create stack
2 to dispose stack
3 to print top
4 to push
5 to pop
6 top and pop
2

enter 0 to exit
1 to create stack
2 to dispose stack
3 to print top
4 to push
5 to pop
6 top and pop
0
navin@DESKTOP-D400C38:/mnt/d/ds lab$
```

```
navin@DESKTOP-D400C38: /mnt/d/ds lab$ ./stk
enter 0 to exit
1 to create stack
2 to dispose stack
3 to print top
4 to push
5 to pop
6 top and pop
1

enter no. of elements
5

enter 0 to exit
1 to create stack
2 to dispose stack
3 to print top
4 to push
5 to pop
6 top and pop
4

enter element to push
1

enter 0 to exit
1 to create stack
2 to dispose stack
3 to print top
4 to push
5 to pop
6 top and pop
4

enter element to push
2

enter 0 to exit
1 to create stack
2 to dispose stack
3 to print top
4 to push
5 to pop
6 top and pop
4

enter element to push
3
```

Queue

Source code

```
#include<stdio.h>
#include<stdlib.h>
#define MIN_QUEUE_SIZE 2

struct queue_record
{
    int capacity;
    int front;
    int rear;
    int *qarr;
};

typedef struct queue_record *queue;

queue create_queue(int max_elements )
{
    queue q;

    if( max_elements < MIN_QUEUE_SIZE )
        printf("queue size is too small");

    q = (struct queue_record *) malloc( sizeof( struct queue_record ) )
;

    if( q == NULL )
        printf("Out of space!!!");

    q->qarr = (int *)malloc( sizeof( int ) * max_elements );

    if( q->qarr == NULL )
        printf("Out of space!!!");

    q-> front = q->rear = -1;
    q->capacity = max_elements;
    return( q );
}
```

```

// Free the memory
void dispose(queue q)
{
    if(q == NULL)
        printf(" queue is not available");
    else
    {
        free(q->qarr);
        free(q);
    }
}

int isEmpty(queue q)
{
    return (q->front == -1 && q->rear == -1 );
}

void make_empty(queue q)
{
    if( !isEmpty(q) )
        q->front = q->rear = -1 ;
    else
        printf(" Already empty");
}

int isFull(queue q)
{
    return( q->rear == q->capacity -1 );
}

void enqueue( queue q , int x)
{
    if( !isFull(q) )
    {
        q->qarr[ ++q->rear ] = x ;           // ++q->rear ; q->qarr[q-
>rear] = x ;
        if ( q->front == -1)
            q->front = 0;
    }
    else
        printf( "\n overflow \n ");
}

void dequeue( queue q)
{

```

```

    if( !isEmpty(q) )
    {
        if( q->front == q->rear )
            q->front = q->rear = -1 ;    // make_empty(q);
        else
            q->front ++ ;
    }
    else
        printf("\n underflow \n ");
}

int front(queue q)
{
    if( ! isEmpty(q) )
    {
        printf("\n front element %d \n ", q->qarr[q->front] );
    }
    else
    {
        printf("\n empty \n ");
    }
}

int front_and_dequeue( queue q)
{
    if( !isEmpty(q) )
    {
        front(q);
        dequeue(q);
    }
    else
    {
        printf("\n empty and underflow \n ");
    }
}

int main()

```

```

{
int c;
queue q;
printf("\nenter 0 to exit\n 1 to create queue \n 2 to dispose queue \n 3 to make empty \n 4 to enqueue \n 5 to dequeue\n 6 front and deque \n 7 front \n");
scanf("%d",&c);
do
{
switch(c)
{
case 0:{

printf("\nexiting\n");
break;

}

case 1:{

int me;printf("\nenter no. of elements \n");
scanf("%d",&me);
q=create_queue(me);

break;

}

case 2:{

dispose(q);
break;

}

case 3:{

make_empty(q);

break;

}

case 4:{

int n;
printf("\n enter element to push \n");
scanf("%d",&n);
enqueue(q,n);
break;

}

case 5:{

dequeue(q);

```

```

        break;
    }
    case 6:{
        front_and_dequeue(q);

        break;
    }
    case 7:{
        front(q);

        break;
    }
    default: {
        printf("\n only 0 to 5\n");
        break;}

}

    if(c==0) break;

    printf("\nenter 0 to exit\n 1 to create queue \n 2 to dispose queue
\n 3  to make empty \n 4 to enqueue \n 5 to dequeue\n 6 front and deque \n 7 fro
nt \n");
    scanf("%d",&c);

}while(c!=0);
return 0;

}

```

Code screenshots

The screenshot shows the Visual Studio Code editor with a C program. The Explorer sidebar on the left shows a file named 'lq.c' under the 'D:\> ds lab' directory. The main editor area displays the following code:

```
150 dispose(q);
151 break;
152
153 case 3:{
154     make_empty(q);
155     break;
156
157 case 4:{
158     int n;
159     printf("\n enter element to push \n");
160     scanf("%d",&n);
161     enqueue(q,n);
162     break;
163
164 case 5:{
165     dequeue(q);
166     break;
167
168 case 6:{
169     front_and_dequeue(q);
170     break;
171
172 case 7:{
173     front(q);
174     break;
175
176 default: {
177     printf("\n only 0 to 5\n");
178     break;
179
180
181
182
183
184
185
186
187
```

The status bar at the bottom indicates 'Ln 191, Col 32', 'Spaces: 4', 'UTF-8', 'CRLF', 'C', 'Win32', and the date '07-10-2020'.

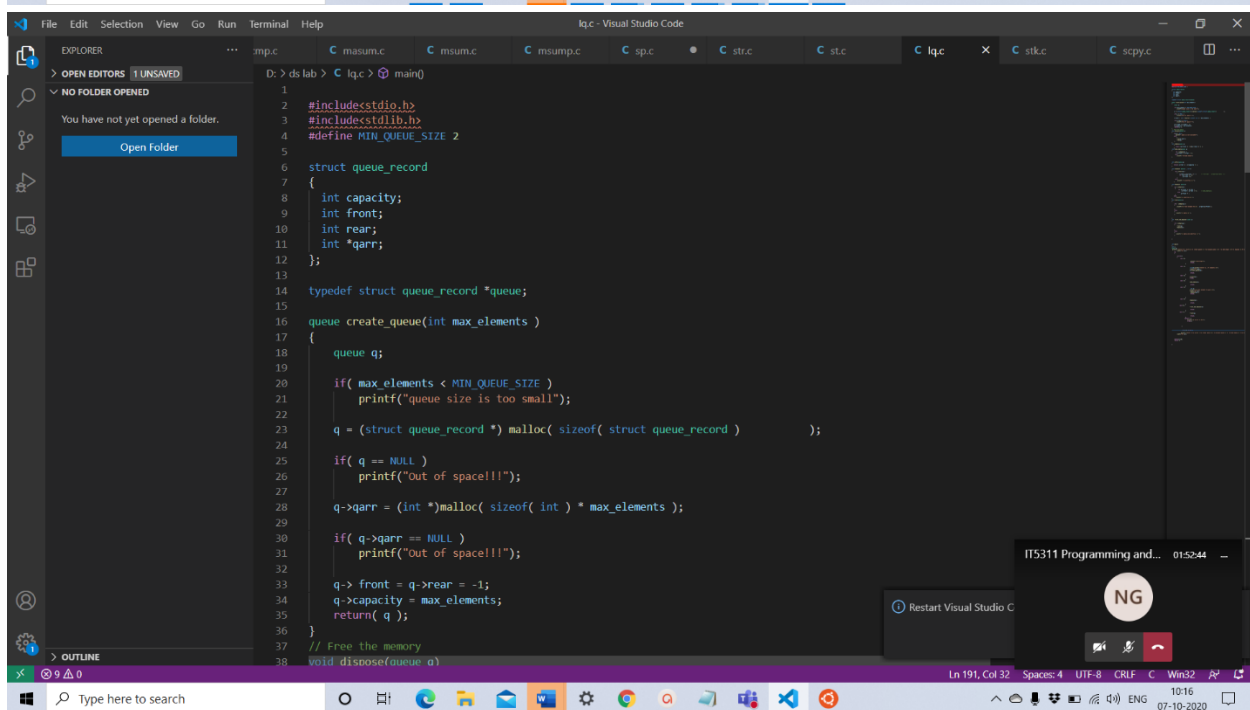
The screenshot shows the Visual Studio Code editor with the same C program. The main function is visible, showing the initialization of variables and the start of the switch statement.

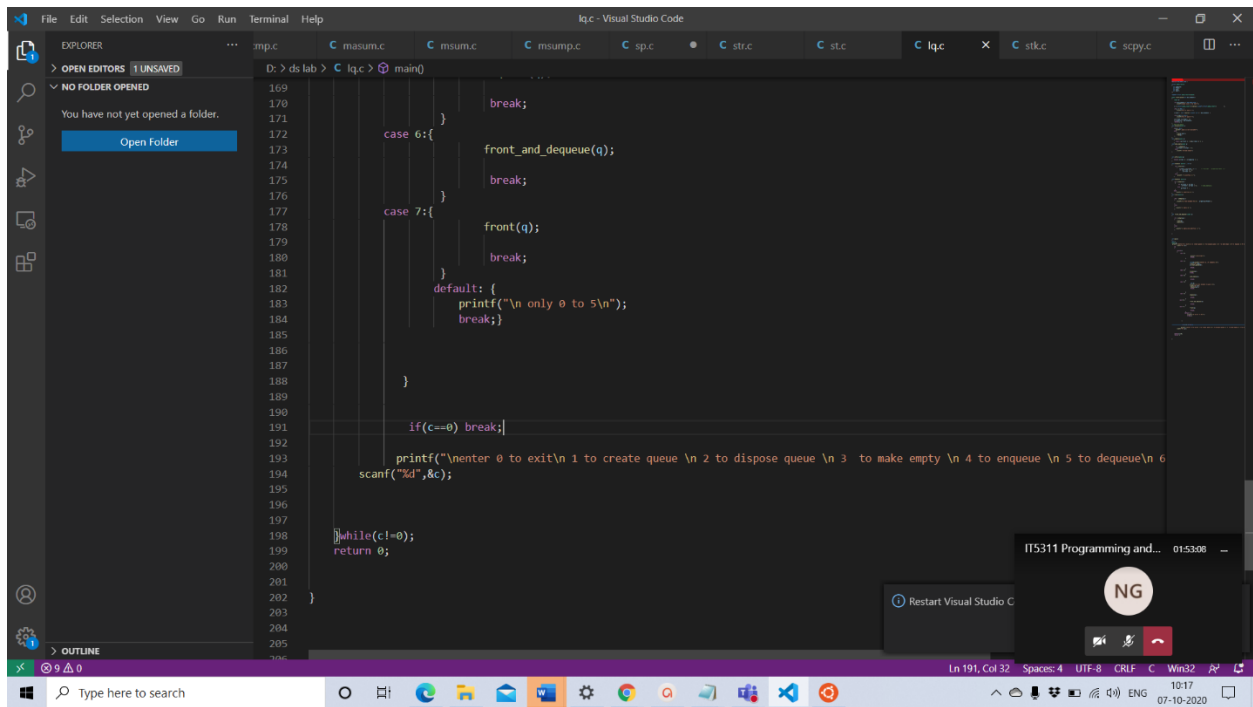
```
120
121 }
122
123
124
125 int main()
126 {
127     int c;
128     queue q;
129     printf("\n enter 0 to exit\n 1 to create queue \n 2 to dispose queue \n 3 to make empty \n 4 to enqueue \n 5 to dequeue \n 6 front and de
130     scanf("%d",&c);
131     do
132     {
133         switch(c)
134         {
135             case 0:{
136                 printf("\nexiting\n");
137                 break;
138             }
139
140             case 1:{
141                 int me;printf("\n enter no. of elements \n");
142                 scanf("%d",&me);
143                 q=create_queue(me);
144                 break;
145             }
146
147             case 2:{
148                 dispose(q);
149                 break;
150             }
151
152             case 3:{
153                 make_empty(q);
154                 break;
155
156
157
```

The status bar at the bottom indicates 'Ln 191, Col 32', 'Spaces: 4', 'UTF-8', 'CRLF', 'C', 'Win32', and the date '07-10-2020'.


```
85         else
86             q->front ++ ;
87     }
88     else
89         printf("\n underflow \n ");
90 }
91 int front(queue q)
92 {
93     if( ! isEmpty(q) )
94     {
95         printf("\n front element %d \n ", q->qarr[q->front] );
96     }
97     else
98     {
99         printf("\n empty \n ");
100     }
101 }
102
103
104
105
106 int front_and_dequeue( queue q)
107 {
108     if( isEmpty(q) )
109     {
110         front(q);
111         dequeue(q);
112     }
113     else
114     {
115         printf("\n empty and underflow \n ");
116     }
117 }
118
119
120
121
122
```

```
59
60
61 int isFull(queue q)
62 {
63     return( q->rear == q->capacity -1 );
64 }
65
66
67 void enqueue( queue q , int x)
68 {
69     if( isEmpty(q) )
70     {
71         q->qarr[ ++q->rear ] = x ;      // ++q->rear ; q->qarr[q->rear] = x ;
72         if ( q->front == -1)
73             q->front = 0;
74     }
75     else
76         printf( "\n overFlow \n ");
77 }
78
79 void dequeue( queue q)
80 {
81     if( isEmpty(q) )
82     {
83         if( q->front == q->rear )
84             q->front = q->rear = -1 ;    // make_empty(q);
85         else
86             q->front ++ ;
87     }
88     else
89         printf("\n underflow \n ");
90 }
91 int front(queue q)
92 {
93     if( ! isEmpty(q) )
94     {
95         printf("\n front element %d \n ", q->qarr[q->front] );
96     }
97 }
```





Output screenshots

```
navin@DESKTOP-D400C38:/mnt/d/ds lab$ ./lq
enter 0 to exit
1 to create queue
2 to dispose queue
3 to make empty
4 to enqueue
5 to dequeue
6 front and deque
7 front
1
enter no. of elements
5
enter 0 to exit
1 to create queue
2 to dispose queue
3 to make empty
4 to enqueue
5 to dequeue
6 front and deque
7 front
3
Already empty
enter 0 to exit
1 to create queue
2 to dispose queue
3 to make empty
4 to enqueue
5 to dequeue
6 front and deque
7 front
4
enter element to push
1
enter 0 to exit
1 to create queue
2 to dispose queue
3 to make empty
4 to enqueue
5 to dequeue
6 front and deque
7 front
4
enter element to push
2
```

```
navin@DESKTOP-D400C38:/mnt/d/ds lab$ ./lq
enter element to push
2
enter 0 to exit
1 to create queue
2 to dispose queue
3 to make empty
4 to enqueue
5 to dequeue
6 front and deque
7 front
4
enter element to push
3
enter 0 to exit
1 to create queue
2 to dispose queue
3 to make empty
4 to enqueue
5 to dequeue
6 front and deque
7 front
4
enter element to push
4
enter 0 to exit
1 to create queue
2 to dispose queue
3 to make empty
4 to enqueue
5 to dequeue
6 front and deque
7 front
4
enter element to push
5
enter 0 to exit
1 to create queue
2 to dispose queue
3 to make empty
4 to enqueue
5 to dequeue
6 front and deque
```

ITS311 Programming and... 01:51:15

NG

📹 🔊 📞

```
navin@DESKTOP-D400C38: /mnt/d/ds lab
enter element to push
5

enter 0 to exit
1 to create queue
2 to dispose queue
3 to make empty
4 to enqueue
5 to dequeue
6 front and deque
7 front
4

enter element to push
6

overflow

enter 0 to exit
1 to create queue
2 to dispose queue
3 to make empty
4 to enqueue
5 to dequeue
6 front and deque
7 front
7

front element 1

enter 0 to exit
1 to create queue
2 to dispose queue
3 to make empty
4 to enqueue
5 to dequeue
6 front and deque
7 front
5

enter 0 to exit
1 to create queue
2 to dispose queue
3 to make empty
4 to enqueue
5 to dequeue
6 front and deque
7 front
7

IT5311 Programming and... 01:51:21 --
NG
[Microphone Icon] [Camera Icon] [End Call Icon]
```

```
navin@DESKTOP-D400C38: /mnt/d/ds lab
enter 0 to exit
1 to create queue
2 to dispose queue
3 to make empty
4 to enqueue
5 to dequeue
6 front and deque
7 front
7

front element 2

enter 0 to exit
1 to create queue
2 to dispose queue
3 to make empty
4 to enqueue
5 to dequeue
6 front and deque
7 front
6

front element 2

enter 0 to exit
1 to create queue
2 to dispose queue
3 to make empty
4 to enqueue
5 to dequeue
6 front and deque
7 front
7

front element 3

enter 0 to exit
1 to create queue
2 to dispose queue
3 to make empty
4 to enqueue
5 to dequeue
6 front and deque
7 front
5

enter 0 to exit
1 to create queue
2 to dispose queue
3 to make empty

IT5311 Programming and... 01:51:27 --
NG
[Microphone Icon] [Camera Icon] [End Call Icon]
```

```
navin@DESKTOP-D400C38: /mnt/d/ds lab
enter 0 to exit
1 to create queue
2 to dispose queue
3 to make empty
4 to enqueue
5 to dequeue
6 front and deque
7 front
7
front element 4
enter 0 to exit
1 to create queue
2 to dispose queue
3 to make empty
4 to enqueue
5 to dequeue
6 front and deque
7 front
6
front element 4
enter 0 to exit
1 to create queue
2 to dispose queue
3 to make empty
4 to enqueue
5 to dequeue
6 front and deque
7 front
6
front element 5
enter 0 to exit
1 to create queue
2 to dispose queue
3 to make empty
4 to enqueue
5 to dequeue
6 front and deque
7 front
6
empty and underflow
enter 0 to exit
1 to create queue
```

```
navin@DESKTOP-D400C38: /mnt/d/ds lab
5 to dequeue
6 front and deque
7 front
6
empty and underflow
enter 0 to exit
1 to create queue
2 to dispose queue
3 to make empty
4 to enqueue
5 to dequeue
6 front and deque
7 front
5
underflow
enter 0 to exit
1 to create queue
2 to dispose queue
3 to make empty
4 to enqueue
5 to dequeue
6 front and deque
7 front
2
enter 0 to exit
1 to create queue
2 to dispose queue
3 to make empty
4 to enqueue
5 to dequeue
6 front and deque
7 front
0
navin@DESKTOP-D400C38: /mnt/d/ds lab$
```