

LAB 14/10/20

SWAMINATHAN NAVINASHOK

2019115126

CIRCULAR QUEUE

SOURCE CODE

```
#include<stdio.h>
#include<stdlib.h>
#define MIN_QUEUE_SIZE 5

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

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

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

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

void dequeue( queue q)
{
    if( ! isEmpty(q) )
    {
        if( q->front == q->rear )
            q->front = q->rear = -1 ;      // make_empty(q);
        else
            q->front = (q->front +1 )%(q->capacity) ;
    }
    else
        printf(" underflow");
}

int front(queue q)
{
    return(q->qarr[q->front]);
}

int front_dequeue( queue q)
{
    if( ! isEmpty(q) )

```

```

    {
        if( q->front == q->rear )
            q->front = q->rear = -1 ;    // make_empty(q);
        else
            q->front = (q->front +1 )%q->capacity ;
        return(q->qarr[q->front]);
    }
    else
        printf(" underflow");
}

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

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

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:{
        int n=front_dequeue(q);
        printf("%d",n);

        break;
    }
    case 7:{
        int k=front(q);

        printf("%d",k);

        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 front \n");
    scanf("%d",&c);

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

}
```

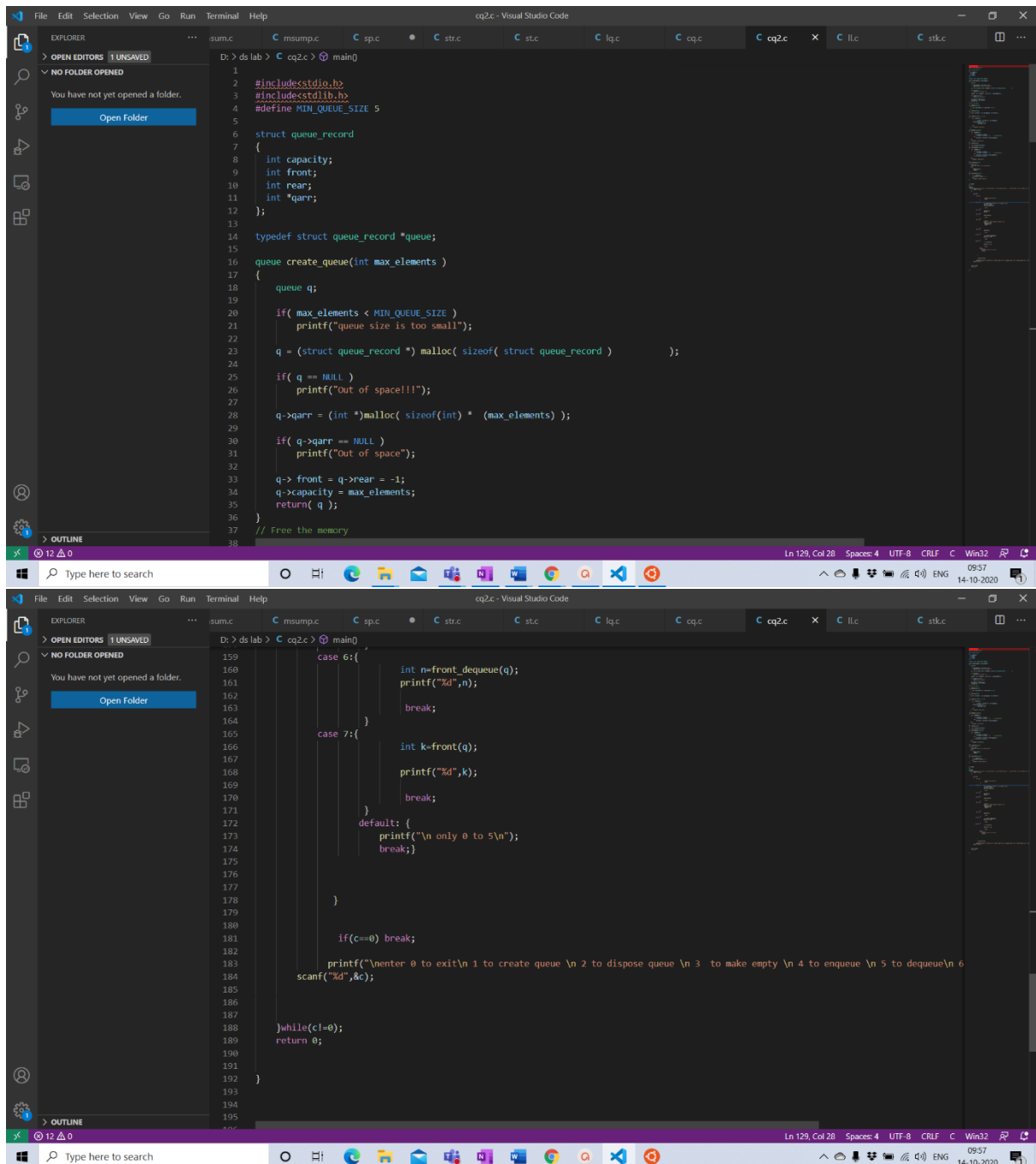
SCREENSHOT OF SOURCE CODE

```
131     scanf("%d",&me);
132     q=create_queue(me);
133
134     break;
135
136     case 2:{
137         dispose(q);
138         break;
139     }
140     case 3:{
141         make_empty(q);
142         break;
143     }
144     case 4:{
145         int n;
146         printf("\n enter element to push \n");
147         scanf("%d",&n);
148         enqueue(q,n);
149         break;
150     }
151     case 5:{
152         dequeue(q);
153         break;
154     }
155     case 6:{
156         int n=front_dequeue(q);
157         printf("%d",n);
158         break;
159     }
160     case 7:{
161         int k=front(q);
162         printf("%d",k);
163     }
164 }
165
166
167
168
```

```
102 void make_empty(queue q)
103 {
104     if( ! isempty(q) )
105         q->front = q->rear = -1 ;
106     else
107         printf(" Already empty");
108 }
109
110
111 int main()
112 {
113     int c;
114     queue q;
115     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
116     scanf("%d",&c);
117     do
118     {
119         switch(c)
120         {
121             case 0:{
122                 printf("\nexiting\n");
123                 break;
124             }
125             case 1:{
126                 int me;printf("\n enter no. of elements \n");
127                 scanf("%d",&me);
128                 q=create_queue(me);
129                 break;
130             }
131             case 2:{
132                 dispose(q);
133                 break;
134             }
135             case 3:{
136                 make_empty(q);
137                 break;
138             }
139         }
140     } while(c != 0);
141 }
```

```
71 else printf(" underflow");
72 }
73 int front(queue q)
74 {
75     return(q->qarr[q->front]);
76 }
77 int front_dequeue( queue q)
78 {
79     if( ! isEmpty(q) )
80     {
81         if( q->front == q->rear )
82             q->front = q->rear = -1 ; // make_empty(q);
83         else
84             q->front = (q->front +1 )%q->capacity ;
85         return(q->qarr[q->front]);
86     }
87     else
88         printf(" underflow");
89 }
90 void dispose(queue q)
91 {
92     if(q == NULL)
93         printf(" queue is not available");
94     else
95     {
96         free(q->qarr);
97         free(q);
98     }
99 }
100 void make_empty(queue q)
101 {
102     if( ! isEmpty(q) )
103         q->front = q->rear = -1 ;
104     else
105         printf(" Already empty");
106 }
107 }
```

```
36 }
37 // Free the memory
38
39 int isEmpty(queue q)
40 {
41     return (q->front == -1 && q->rear == -1 );
42 }
43
44 int isFull(queue q)
45 {
46     return( (q->rear + 1 )% q->capacity == q->front );
47 }
48
49 void enqueue( queue q , int x)
50 {
51     if( ! isFull(q) )
52     {
53         q->rear = (q->rear +1 ) % q->capacity;
54         q->qarr[q->rear ] = x ;
55         if ( q->front == -1)
56             q->front = 0;
57     }
58     else
59         printf( "overflow");
60 }
61
62 void dequeue( queue q)
63 {
64     if( ! isEmpty(q) )
65     {
66         if( q->front == q->rear )
67             q->front = q->rear = -1 ; // make_empty(q);
68         else
69             q->front = (q->front +1 )%(q->capacity) ;
70     }
71     else
72         printf(" underflow");
73 }
```



SCREENSHOT OF OUTPUT

A screenshot of a Windows desktop. The background is a dark blue gradient. In the foreground, there is a large black terminal window with white text. The terminal shows a user named 'navin' at a desktop named 'D400C38' in the directory '/mnt/ds/lab'. The user has entered the command 'enter element to push' and pressed enter, resulting in a blank line. Then, the user entered 'enter 0 to exit' and pressed enter, followed by a list of menu options: '1 to create queue', '2 to dispose queue', '3 to make empty', '4 to enqueue', '5 to dequeue', '6 front and deque', and '7 front'. This sequence of actions was repeated three times. In the bottom right corner of the terminal window, there is a Zoom meeting overlay. The overlay shows the text 'Meeting in "General" 50:58' and a small video thumbnail of a globe. Below the thumbnail are three icons: a speech bubble, a microphone, and a red hang-up button. At the very bottom of the screen is the Windows taskbar, which includes the Start button, a search bar with the text 'Type here to search', and several pinned application icons: File Explorer, Microsoft Edge, Outlook, Teams, Word, PowerPoint, Chrome, and the Zoom application. The system tray on the right side of the taskbar shows the date and time as '14-10-2020 09:20' and the language as 'ENG'.

The image shows a Windows desktop environment. A terminal window is open, displaying a series of commands and their outputs for a queue data structure implementation. The commands include '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', and '7 front'. The output shows the queue being created, disposed, and then elements being added and removed. A Windows Teams meeting overlay is visible in the bottom right corner, titled 'Meeting in "General"' with a duration of 50:54. The taskbar at the bottom shows various application icons, including File Explorer, Edge, and several instances of Word and Excel. The system tray in the bottom right corner shows the date and time as 14-10-2020 and 09:20, along with language and connectivity icons.

The screenshot shows a Windows desktop environment. A terminal window is open, displaying the output of a program that implements a queue using two stacks. The program prompts the user to enter commands to perform various queue operations. The commands entered are: 'enter element to push', '0', '1 to create queue', '2 to dispose queue', '3 to make empty', '4 to enqueue', '5 to dequeue', '6 front and deque', '7 front', and '4'. This sequence is repeated three times. A Zoom meeting window titled 'Meeting in "General"' is also open, showing a video call in progress with a globe icon and controls for video, audio, and chat. The taskbar at the bottom shows various application icons, and the system tray on the right indicates the time as 09:20 on 14-10-2020.

The screenshot shows a Windows 10 desktop environment. A black terminal window titled "navin@DESKTOP-D400C38: /mnt/d/ds lab" is open, displaying a menu-driven program for implementing a queue using two stacks. The menu options are: 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, and 8 enter element to push. The user has entered '2' twice, selecting 'dispose queue'. A Zoom meeting window titled "Meeting in 'General'" is overlaid on the bottom right of the screen, showing a globe icon and controls for video, audio, and mute. The taskbar at the bottom contains various application icons, including File Explorer, Edge, Word, and Chrome. The system tray on the far right shows the time as 09:20 on 14-10-2020.

```
navin@DESKTOP-D40C38: /mnt/d/ds lab
navin@DESKTOP-D40C38: /mnt/d/ds lab$ ./cq2

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
4
queue size is too small
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
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

Meeting in "General" 50:34

navin@DESKTOP-D40C38: /mnt/d/ds lab
2 to dispose queue
3 to make empty
4 to enqueue
5 to dequeue
6 front and deque
7 front
6
6
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
6
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
2
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-D40C38: /mnt/d/ds lab$
```

LINKED LIST

SOURCE CODE

```

#include<stdio.h>
#include<stdlib.h>
struct noderec
{
    int info;
    struct noderec *next ;
};
typedef struct noderec * node ;
node start, last, new, temp, prev ;
void create()
{
    start = last = NULL ;
}
int isEmpty()
{
    return start ==NULL ;
}
void insertEnd(int x)
{
    new = (struct noderec*)malloc(sizeof(struct noderec) );
    new -> info = x;
    new -> next = NULL;
    if( isEmpty())
        start= last= new ;
    else
    {
        last->next = new ;
        last = new ;
    }
}

void insertBeg(int x)
{
    new = (struct noderec*)malloc(sizeof(struct noderec) );
    new -> info = x;
    if(isEmpty())
    {
        new->next = NULL ;
        start = last = new ;
    }
    else
    {
        new->next = start ;
        start = new ;
    }
}

void insertMid(int x, int pos)

```

```

{
    int i = 1;
    new = (struct noderec*)malloc(sizeof(struct noderec) );
    new -> info = x;
    if( isEmpty())
    {
        new -> next = NULL;
        start = last = new;
    }
    else
    {
        temp = start ;
        while( temp != NULL && i<pos-1)
        {
            temp = temp->next;
            i++ ;
        }
        if( pos == 1)
        {
            new-> next = start ;
            start = new ;
        }
        else if ( temp == NULL)
        {
            new -> next = NULL;
            last -> next = new;
            last = new ;
        }
        else
        {
            new -> next = temp -> next ;
            temp->next = new ;
        }
    }
}

```

```

void display()
{
    if( ! isEmpty())
    {
        temp = start ;
        while( temp != NULL)
        {
            printf("%d \n", temp->info );
            temp = temp ->next ;
        }
    }
    else

```

```

        printf("list is empty");
    }

void delete( int x)
{
    prev = start;
    temp= start;
    if(isEmpty())
    {
        printf("list is empty");
    }
    else
    {
        while( temp != NULL  &&  temp->info != x)
        {
            prev= temp;
            temp = temp->next ;
        }
        if(temp == NULL)
        {
            printf(" no such element found");
        }
        else
        {
            if( start-> info ==x)    // if(temp==start)
            {
                if( start == last)
                {
                    start = NULL ;
                    free(temp) ;
                }
                else
                {
                    start = start ->next;
                    free(temp);
                }
            }
            else if(temp == last)
            {
                prev->next = temp->next; // prev->next = NULL
                last = prev;
                free(temp);
            }
            else
            {
                prev-> next = temp->next;
                free(temp);
            }
        }
    }
}

```

```

    }
}

void deletelist()
{
    if(!isEmpty())
    {
        temp = start ;
        while(temp !=NULL)
        {
            prev = temp ;
            temp = temp->next;
            free(prev);
        }
        start = NULL;
    }
    else
    {
        printf("list is empty");
    }
}

int main()
{
    int c,i,pos;
    node n;
    printf("\nenter 1 to create \n 2 to insert at End \n 3  insert at Beg \n 4 in
sert at Mid \n 5 display\n 6 delete \n 7 deletelist \n 0 to exit \n");
    scanf("%d",&c);
    do
    {
        switch(c)
        {
            case 0:{
                printf("\nexiting\n");
                break;
            }

            case 1:{
                create();
                break;
            }

```

```
case 2:{
    printf("\n enter element \n ");
    scanf("%d",&i);
    insertEnd(i);
    break;
}
case 3:{
    printf("\n enter element \n");
    scanf("%d",&i);
    insertBeg(i);

    break;
}
case 4:{
    printf("\n enter element \n");
    scanf("%d",&i);
    printf("\n enter pos \n");
    scanf("%d",&pos);
    insertMid(i,pos);
    break;
}
case 5:{
    printf("\n \n ");
    display();
    printf("\n \n ");

    break;
}
case 6:{
    printf("\n enter element \n");
    scanf("%d",&i);
    delete(i);

    break;
}
case 7:{

    deletelist();

    break;
}

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



```
    }  
    if(c==8) break;  
  
    printf("\nenter 1 to create \n 2 to insert at End \n 3  insert a  
t Beg \n 4 insert at Mid \n 5 display\n 6 delete \n 7 deletelist \n 0 to exit  
\n");  
    scanf("%d",&c);  
  
}while(c!=0);  
return 0;  
  
}
```

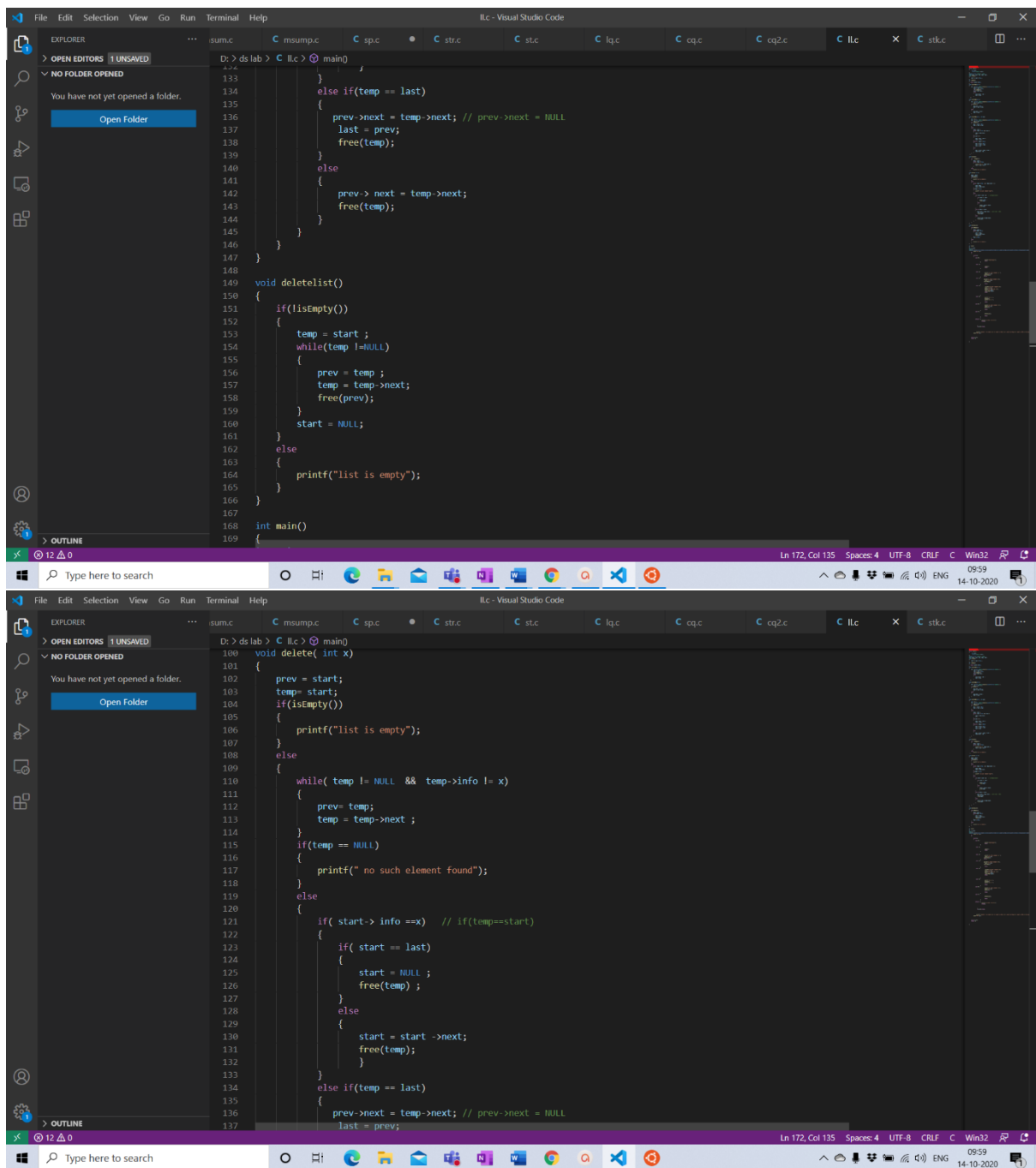
SCREENSHOT OF SOURCE CODE

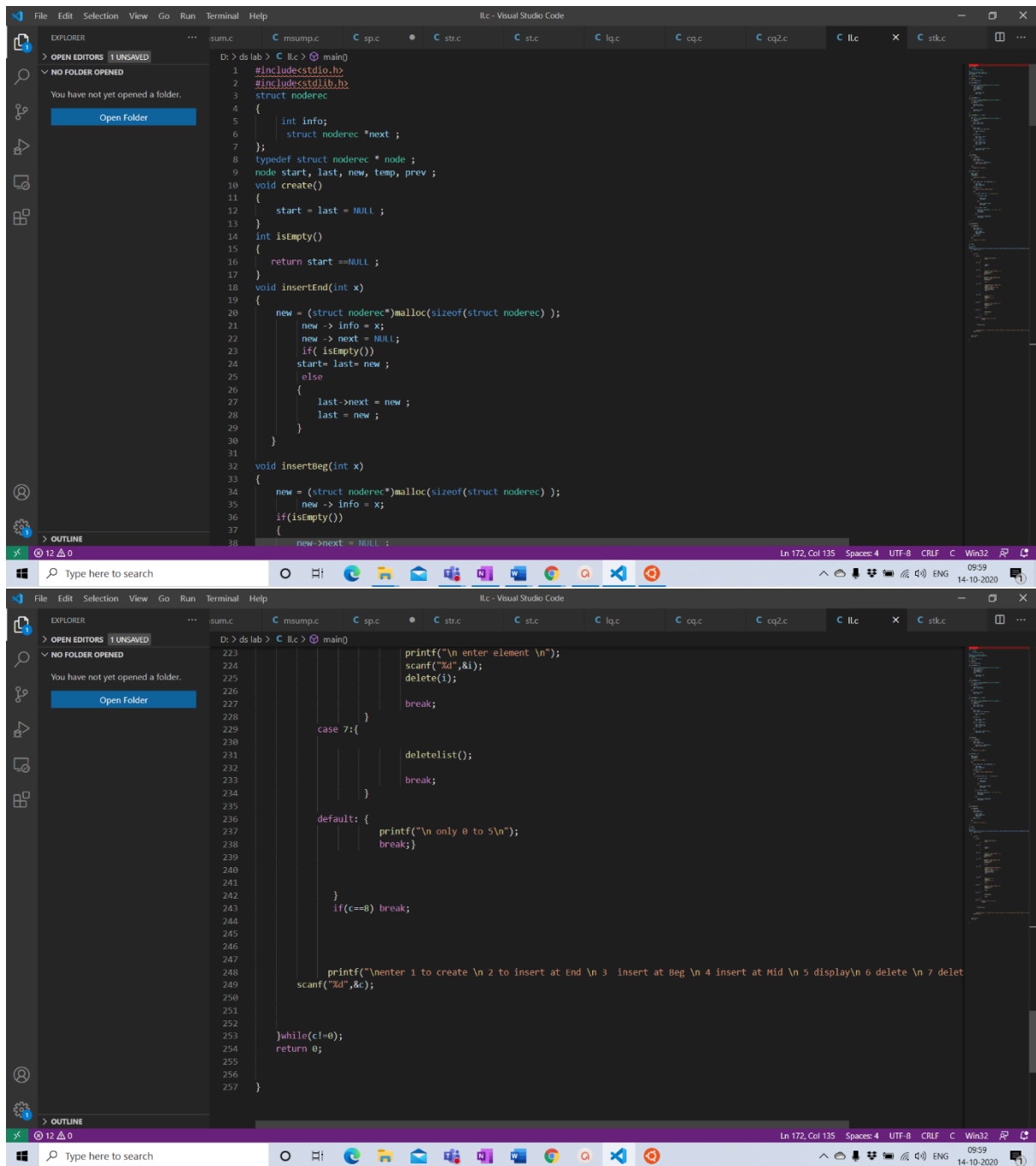
The screenshot shows the Visual Studio Code editor with a C program. The Explorer sidebar on the left shows a file named 'sum.c' and a message 'NO FOLDER OPENED'. The main editor area displays the following code:

```
197     }
198     case 3:{
199         printf("\n enter element \n");
200         scanf("%d",&i);
201         insertBeg(i);
202         break;
203     }
204     case 4:{
205         printf("\n enter element \n");
206         scanf("%d",&i);
207         printf("\n enter pos \n");
208         scanf("%d",&pos);
209         insertMid(i,pos);
210         break;
211     }
212     case 5:{
213         printf("\n \n ");
214         display();
215         printf("\n \n ");
216         break;
217     }
218     case 6:{
219         printf("\n enter element \n");
220         scanf("%d",&i);
221         delete(i);
222         break;
223     }
224     case 7:{
225         deletelist();
226         break;
227     }
228 }
```

The screenshot shows the Visual Studio Code editor with the same C program, displaying the main function. The Explorer sidebar on the left shows a file named 'sum.c' and a message 'NO FOLDER OPENED'. The main editor area displays the following code:

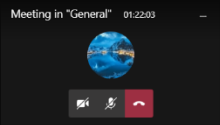
```
166     }
167 }
168 int main()
169 {
170     int c,i,pos;
171     node n;
172     printf("\nenter 1 to create \n 2 to insert at End \n 3 insert at Beg \n 4 insert at Mid \n 5 display\n 6 delete \n 7 deletelist \n 0 to exit\n");
173     scanf("%d",&c);
174     do
175     {
176         switch(c)
177         {
178             case 0:{
179                 printf("\nexiting\n");
180                 break;
181             }
182             case 1:{
183                 create();
184                 break;
185             }
186             case 2:{
187                 printf("\n enter element \n ");
188                 scanf("%d",&i);
189                 insertEnd(i);
190                 break;
191             }
192             case 3:{
193                 printf("\n enter element \n");
194                 scanf("%d",&i);
195                 insertBeg(i);
196                 break;
197             }
198             case 4:{
199                 printf("\n enter element \n");
200                 scanf("%d",&i);
201                 insertMid(i,pos);
202                 break;
203             }
204             case 5:{
205                 display();
206                 break;
207             }
208             case 6:{
209                 delete(i);
210                 break;
211             }
212             case 7:{
213                 deletelist();
214                 break;
215             }
216             default:
217                 printf("\n invalid choice\n");
218                 break;
219             }
220         }
221     } while(c != 0);
222 }
```





SCREENSHOT OF OUTPUT

```
navin@DESKTOP-D400C38: /mnt/d/ds lab
enter 1 to create
2 to insert at End
3 insert at Beg
4 insert at Mid
5 display
6 delete
7 deletelist
0 to exit
5
enter element
1
enter 1 to create
2 to insert at End
3 insert at Beg
4 insert at Mid
5 display
6 delete
7 deletelist
0 to exit
5
5
2
2
2
enter 1 to create
2 to insert at End
3 insert at Beg
4 insert at Mid
5 display
6 delete
7 deletelist
0 to exit
7
enter 1 to create
2 to insert at End
3 insert at Beg
4 insert at Mid
5 display
6 delete
7 deletelist
0 to exit
8
```



```
navin@DESKTOP-D400C38: /mnt/d/ds lab
enter element
2
enter 1 to create
2 to insert at End
3 insert at Beg
4 insert at Mid
5 display
6 delete
7 deletelist
0 to exit
3
enter element
5
enter 1 to create
2 to insert at End
3 insert at Beg
4 insert at Mid
5 display
6 delete
7 deletelist
0 to exit
4
enter element
7
enter pos
3
enter 1 to create
2 to insert at End
3 insert at Beg
4 insert at Mid
5 display
6 delete
7 deletelist
0 to exit
5
5
2
7
1
2
```

