

DATE:-21/09/22

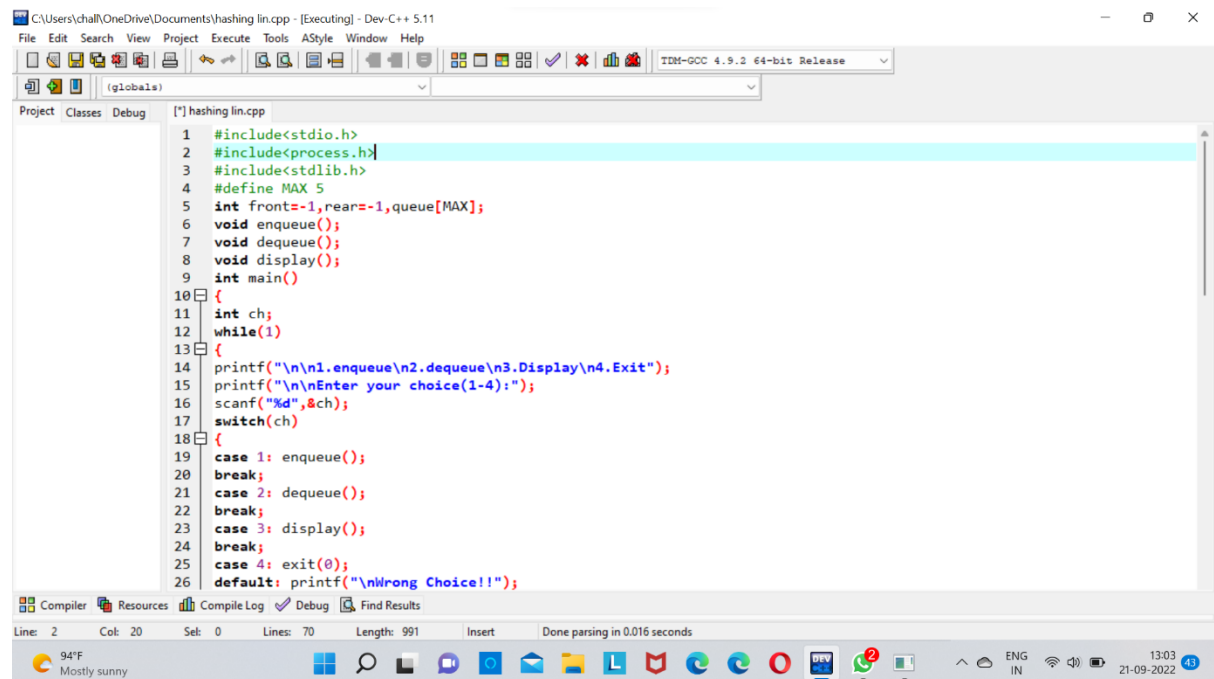
COURSE NAME:-DATA STRUCTURES FOR EXPRESSION EVALUATION

COURSE CODE:-CSA0374

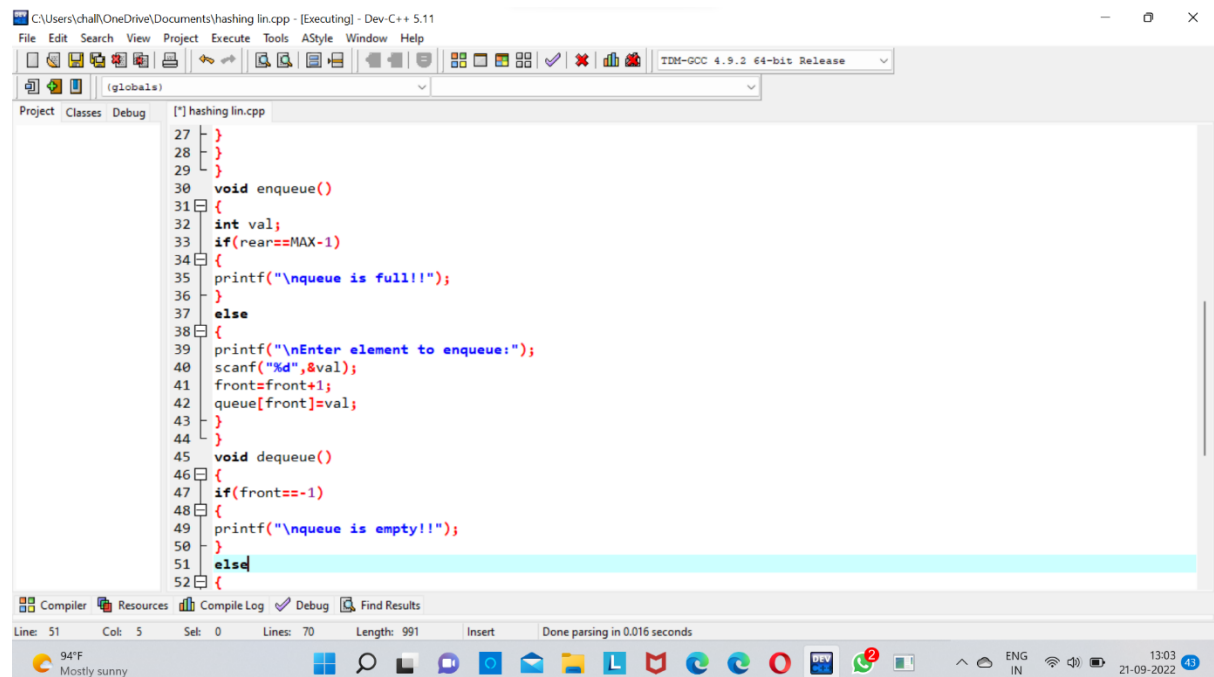
NAME OF THE STUDENT:-USHA NANDHINI .B

REGNO:-192111159

EXPERIMENT:12(Queue,Deque)



```
1 #include<stdio.h>
2 #include<process.h>
3 #include<stdlib.h>
4 #define MAX 5
5 int front=-1,rear=-1,queue[MAX];
6 void enqueue();
7 void dequeue();
8 void display();
9 int main()
10 {
11     int ch;
12     while(1)
13     {
14         printf("\n\n1.enqueue\n2.dequeue\n3.Display\n4.Exit");
15         printf("\n\nEnter your choice(1-4):");
16         scanf("%d",&ch);
17         switch(ch)
18         {
19             case 1: enqueue();
20             break;
21             case 2: dequeue();
22             break;
23             case 3: display();
24             break;
25             case 4: exit(0);
26             default: printf("\nWrong Choice!!");
```



```
27 }
28 }
29 }
30 void enqueue()
31 {
32     int val;
33     if(rear==MAX-1)
34     {
35         printf("\nqueue is full!!");
36     }
37     else
38     {
39         printf("\nEnter element to enqueue:");
40         scanf("%d",&val);
41         front=front+1;
42         queue[front]=val;
43     }
44 }
45 void dequeue()
46 {
47     if(front==-1)
48     {
49         printf("\nqueue is empty!!");
50     }
51     else
52     {
```

The screenshot shows the Dev-C++ IDE with the file `hashing_lin.cpp` open. The code implements a queue using an array with `front` and `rear` pointers. The `display` function prints the elements from `front` to `rear`. The execution window shows the following output:

```

Enter your choice(1-4):1
Enter element to enqueue:4
1.enqueue
2.dequeue
3.Display
4.Exit
Enter your choice(1-4):2
Deleted element is 0
1.enqueue
2.dequeue
3.Display
4.Exit
Enter your choice(1-4):3
queue is empty!!
1.enqueue
2.dequeue
3.Display
4.Exit
Enter your choice(1-4):1

```

EXPERIMENT :-13(HASHING USING LINEARPROBING)

The screenshot shows the Dev-C++ IDE with the file `hashing_lin.cpp` open. The code implements a hash table using linear probing. The `insert` function searches for an empty slot starting from `hkey` and wraps around the table size if necessary. The `search` function finds the index of a given key. The execution window shows the following output:

```

1
enter a value to insert into hash table
22
Press 1. Insert 2. Display 3. Search 4.Exit
2
elements in the hash table are
at index 0 value = 0
at index 1 value = 0
at index 2 value = 22
at index 3 value = 0
at index 4 value = 0
at index 5 value = 0
at index 6 value = 0
at index 7 value = 0
at index 8 value = 0
at index 9 value = 0

```

```
27 scanf("%d",&key);
28 hkey=key*TABLE_SIZE;
29 for(i=0;i<TABLE_SIZE; i++)
30 {
31     index=(hkey+i)*TABLE_SIZE;
32     if(h[index]==key)
33     {
34         printf("value is found at index %d",index);
35         break;
36     }
37 }
38 if(i == TABLE_SIZE)
39     printf("\n value is not found\n");
40
41 void display()
42 {
43     int i;
44     printf("\nelements in the hash table are \n");
45     for(i=0;i< TABLE_SIZE; ++i)
46         printf("\nat index %d \t value = %d",i,h[i]);
47 }
48 main()
49 {
50     int opt,i;
51     while(1)
52     {
53         printf("\nPress 1. Insert \t 2. Display \t 3. Search \t 4.Exit \n");
54         scanf("%d",&opt);
55         switch(opt)
56         {
57             case 1:
58                 insert();
59                 break;
60             case 2:
61                 display();
62                 break;
63             case 3:
64                 search();
65                 break;
66             case 4:exit(0);
67         }
68     }
69 }
```

1
enter a value to insert into hash table
22
Press 1. Insert 2. Display 3. Search 4.Exit
2
elements in the hash table are
at index 0 value = 0
at index 1 value = 0
at index 2 value = 22
at index 3 value = 0
at index 4 value = 0
at index 5 value = 0
at index 6 value = 0
at index 7 value = 0
at index 8 value = 0
at index 9 value = 0
Press 1. Insert 2. Display 3. Search 4.Exit
3
enter search element
42
value is not found
Press 1. Insert 2. Display 3. Search 4.Exit

```
44 printf("\nelements in the hash table are \n");
45 for(i=0;i< TABLE_SIZE; ++i)
46     printf("\nat index %d \t value = %d",i,h[i]);
47 }
48 int main()
49 {
50     int opt,i;
51     while(1)
52     {
53         printf("\nPress 1. Insert \t 2. Display \t 3. Search \t 4.Exit \n");
54         scanf("%d",&opt);
55         switch(opt)
56         {
57             case 1:
58                 insert();
59                 break;
60             case 2:
61                 display();
62                 break;
63             case 3:
64                 search();
65                 break;
66             case 4:exit(0);
67         }
68     }
69 }
```

EXPERIMENT:-14(INSERTION SORT)

