

Name

- Std

Sec

Roll No. _____ Subject.

School/College

School/College Tel. No. .

- Parents Tel. No.

LAB PROGRAM-3

W.A.P. to simulate the working of a queue of integers using an array. Provide the following operations:
 Insert, Delete, Display the program should print appropriate messages for queue.
 Empty & queue overflow conditions.

Algorithm :

INITIALISE

- > Set front = -1
- > Set rear = -1
- > Define array queue [MAX]

INSERT

- > If rear = MAX - 1
 Print ("Queue overflow")
 Exit from function
- > If No new item
 Set front = 0
- > Set rear = rear + 1
- > Get queue [rear] = item
- > Print ("item inserted")

DELETE

- > If front = -1 & rear = -1
 Print ("queue is empty")
 Exit from function
- > Set item = queue [front]
- > If front = rear
 Set front = rear = -1
 Print ("queue is empty")

→ If front > rear

Set front++ pointers with character - of queue
Print item (front) if not empty. (works as pointer
taking character out from left, front, rear)

~~DISPLAY~~

→ If front = -1.

Print ("queue is empty")

Exit from function

→ For i from front to rear

Print (queue[i]) (XAM) -> empty queue output

MAIN

→ Repeat until user exits ("quit or exit") first

→ Print insert, delete, display them with func

or fun switch case for all cases and then call the
functions.

Code:

```
#include <stdio.h>
```

```
#include <stdlib.h> until -> (XAM) empty func
```

```
#define N 5
```

```
int front = -1; l = rear + 1 -> front FE
```

```
int rear = -1; ("if less than 0") then
```

```
int queue[N]; (initially most func)
```

```
void enqueue(int n){
```

```
if (rear == N-1) { l = rear - front FE
```

```
printf("queue overflow");
```

```

else if (front == -1 & & rear == -1) { (if queue is empty)
    front = rear = 0; (initialization)
    queue[rear] = x; ("102", "101") (push)
} } (x) mapping
} (empty)
} (Queue is empty : 5910)

void dequeue() { (dequeue)
    if (front == -1 & & rear == -1) {
        printf("queue is empty!"); (if empty)
    }
    else if (front == rear) {
        front = rear = -1; (dequeue : if empty)
    }
    else { ("works b,front") (front); (if not empty)
        printf("deleted element = %d \n", queue[front]);
        front++;
    }
}
}

```

```
void display() {  
    int i;  
    for(int i = front; i < rear++; i++) {  
        printf("%d\n", queue[i]);  
    }  
}
```

```
int main () {  
    int ch;  
    printf ("1. Insert\n2. Delete\n3. display\n4. exit\n");  
    L1: cin >> ch;  
    if (ch == 1) {  
        cout << "Enter the element" << endl;  
        cin >> n1;  
        insert(n1);  
    } else if (ch == 2) {  
        cout << "Enter the element" << endl;  
        cin >> n1;  
        deleteNode(n1);  
    } else if (ch == 3) {  
        display();  
    } else if (ch == 4) {  
        cout << "Program Terminated" << endl;  
        exit(0);  
    } else {  
        cout << "Wrong choice" << endl;  
    }  
    cout << "Do you want to continue?" << endl;  
    cout << "Press Y for Yes and N for No" << endl;  
    cin >> ch;  
    if (ch == 'Y' || ch == 'y') {  
        goto L1;  
    } else if (ch == 'N' || ch == 'n') {  
        cout << "Program Terminated" << endl;  
        exit(0);  
    } else {  
        cout << "Wrong choice" << endl;  
    }  
}
```

```
while(1){  
    int x;  
    printf("enter choice\n");  
    scanf("%d", &ch);  
    if(ch==1){  
        // do something  
    } else if(ch==2){  
        // do something  
    } else if(ch==3){  
        // do something  
    } else {  
        // do something  
    }  
}
```

switch(ch) { (b == node && t == treep) ? /* Case 1 */

Case 1: printf("Enter the number:"); break;

scanf("%d", &n); n = (int)n; break;

enqueue(x);

break;

Case 2: dequeue();

break;

} /* (b == node && t == treep) ? */

Case 3: display(); /* menu */

break;

} /* (b == treep && t == treep) ? */

Case 4: return 0; /* b == treep && t == treep */

default: printf("Invalid choice");

((treep == NULL) && (t == treep)) ? /* Case 1 */

}

(t == treep)

return 0;

}

Output:

1. Insert

2. Delete; if node > i; treep = i treep

3. Display((i), menu, "choice")

4. Exit

Enter choice: 2

~~Queue~~ ^{under} after queue is empty

Enter choice: 1

Enter the number: 23

((inserting) & (Element inserted in node treep)) ? /* Case 1 */

Enter choice: 1

Enter the number: 56

Element inserted

Enter choice: 1

((displaying) & (choice == 3)) ? /* Case 3 */

Enter the number: 89; ((i), "choice")

Element inserted

Enter choice : 1

Enter the number : 24

Element inserted

Enter choice : 1

Enter the number : 58

Element inserted

Enter choice : 2

Enter the number : 12

~~Element inserted~~ Queue overflow

Enter choice : 3

23

56

89

74

58

Enter choice : 2

Element deleted = 23

Enter choice

3

56

89

74

58

Enter choice

4

Q.
8/10