# Laboratory 6

1. Questions
   1. Implement the INSERT, DELETE and PRINT operations on queue.
   2. Implement a priority queue using suitable application.
2. Algorithm
   1. **INSERT, DELETE and PRINT operations on queue**

step 1: start

step 2: declare global variable queue[size], front=rear= -1

step 3: make function for enqueue(insert)

3.1 if rear = size-1 : print queue is full

3.2 else: rear++ ; queue[rear]=value

step 4: make function for dequeue(delete)

4.1 if front> rear: queue is empty

4.2 front++; if front=rear: front=rear=-1

step 5: make function for displaying queue

5.1 if rear = -1 : print queue is empty

5.2 else: for i=front; i<=rear; i++ : print queue[i]

step 6: call the function according in the main body

step 7: stop

* 1. **priority queue using suitable application.**

step 1: start

step 2: to insert node according to priority

(input value and priority from the user )

1. allocate new node
2. put in the data
3. if (head->priority > priority) {

temp->next = head;

head = temp; }

1. else:

while (start->next!=NULL && start->next->priority<p) { start = start->next;

}

temp->next=start->next;

start->next = temp;

}

step 3: call function accordingly in the main body

step 4: stop

1. Program



Figure C program to do INSERT, DELETE and PRINT operations on queue

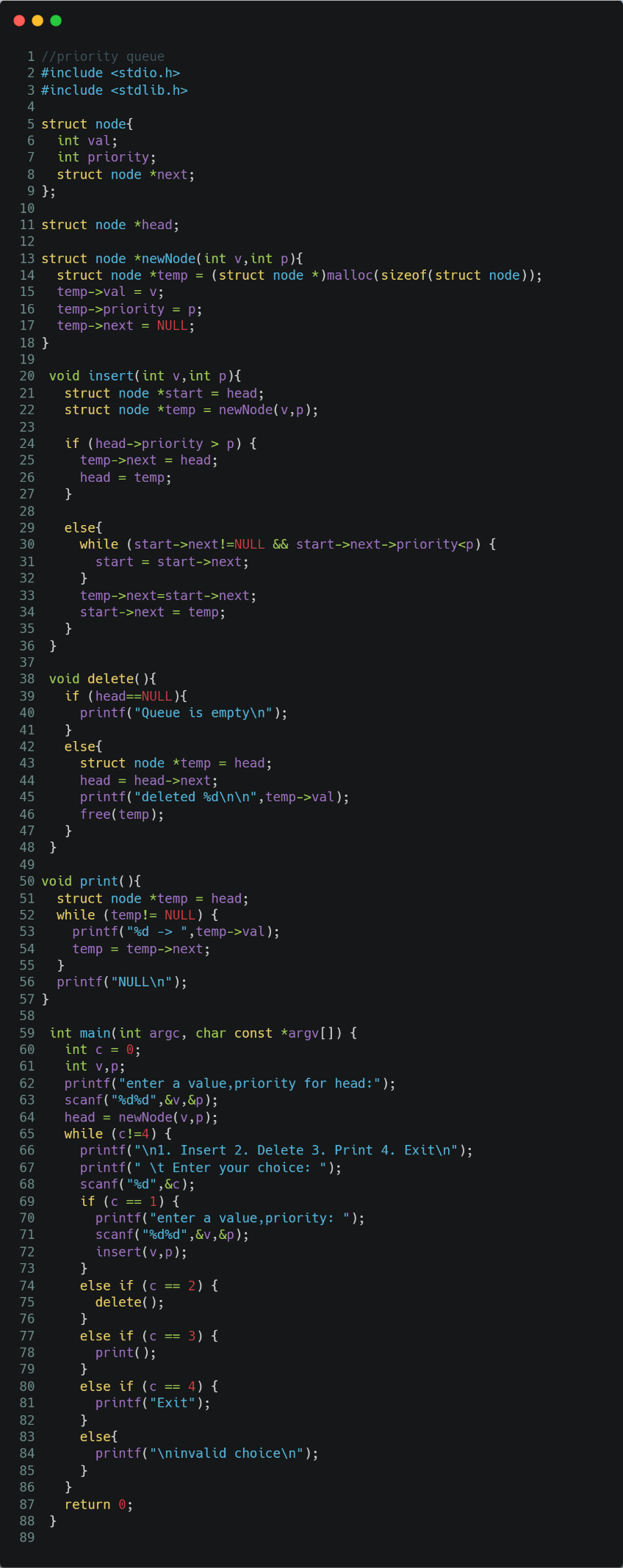
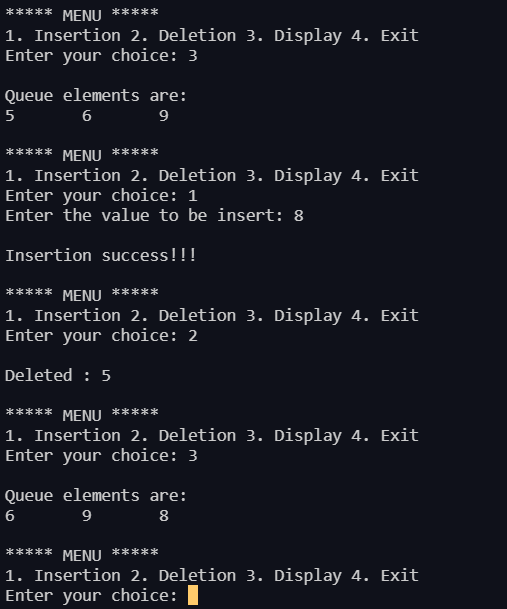
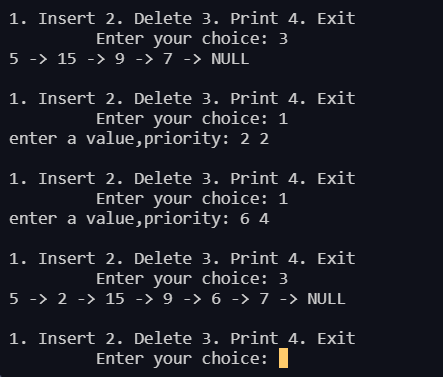


Figure C program to implement priority queue using linked list

1. Presentation of Results





1. Conclusions

Learning Happened:

**Queue** is a linear structure which follows a particular order in which the operations are performed. The order is **F**irst **in** **F**irst **O**ut (**FIFO**).

Queue operations are:

* **Enqueue:**Adds an item to the queue. If the queue is full, then it is said to be an Overflow condition.
* **Dequeue:** Removes an item from the queue. The items are popped in the same order in which they are pushed. If the queue is empty, then it is said to be an Underflow condition.

**Priority** **Queue** is an extension of queue with following properties.

1. Every item has a priority associated with it.
2. An element with high priority is dequeued before an element with low priority.
3. If two elements have the same priority, they are served according to their order in the queue.