Name: SUBHENDU MAJI Roll Number: 18ETCS002121

Laboratory 6

1. Questions

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

2. Algorithm

2.1 INSERT, DELETE and PRINT operations on queue

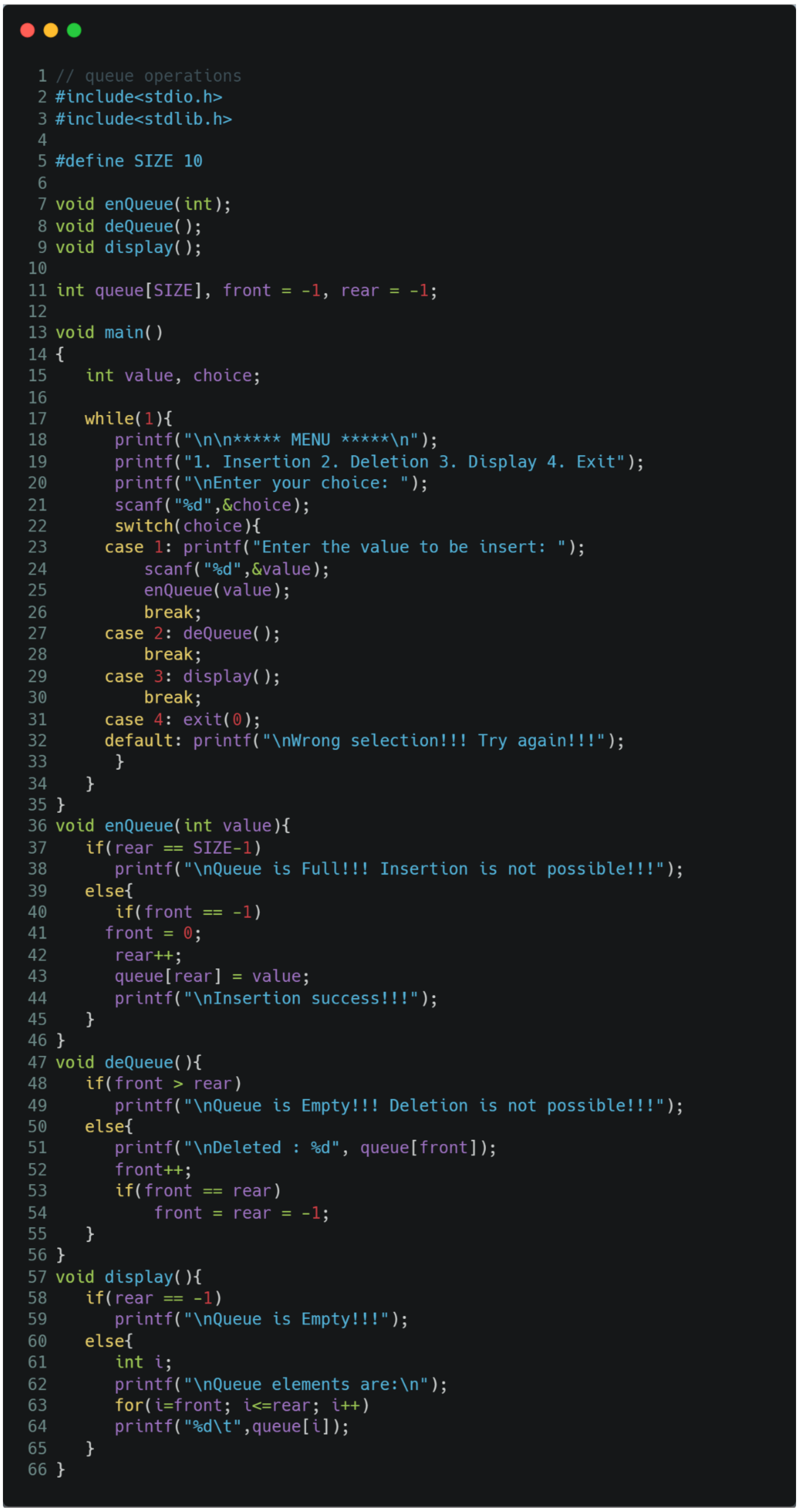
```
step 1: start
step 2: declare global variable queue[size], front=rear= -1
step 3: make function for enqueue(insert)
        3.1 if rear = size-1 : print queue is full
        3.2 else: rear++ ; queue[rear]=value
step 4: make function for dequeue(delete)
        4.1 if front> rear: queue is empty
        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
```

2.2 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; }
    4.     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
```

3. Program

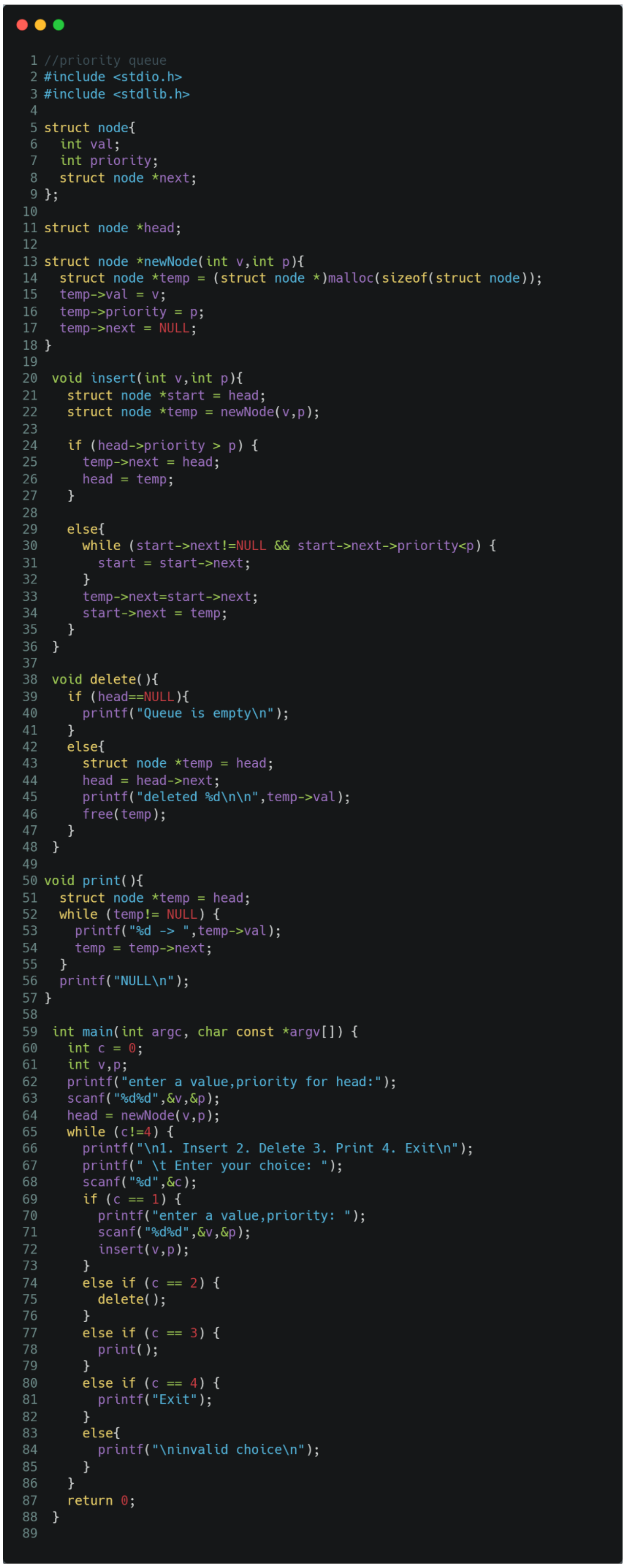
Name: SUBHENDU MAJI Roll Number: 18ETCS002121



```c
// queue operations
#include<stdio.h>
#include<stdlib.h>

#define SIZE 10

void enQueue(int);
void deQueue();
void display();

int queue[SIZE], front = -1, rear = -1;

void main()
{
   int value, choice;

   while(1){
      printf("\n\n***** MENU *****\n");
      printf("1. Insertion 2. Deletion 3. Display 4. Exit");
      printf("\nEnter your choice: ");
      scanf("%d",&choice);
      switch(choice){
     case 1: printf("Enter the value to be insert: ");
         scanf("%d",&value);
         enQueue(value);
         break;
     case 2: deQueue();
         break;
     case 3: display();
         break;
     case 4: exit(0);
     default: printf("\nWrong selection!!! Try again!!!");
      }
   }
}
void enQueue(int value){
   if(rear == SIZE-1)
      printf("\nQueue is Full!!! Insertion is not possible!!!");
   else{
      if(front == -1)
     front = 0;
      rear++;
      queue[rear] = value;
      printf("\nInsertion success!!!");
   }
}
void deQueue(){
   if(front > rear)
      printf("\nQueue is Empty!!! Deletion is not possible!!!");
   else{
      printf("\nDeleted : %d", queue[front]);
      front++;
      if(front == rear)
         front = rear = -1;
   }
}
void display(){
   if(rear == -1)
      printf("\nQueue is Empty!!!");
   else{
      int i;
      printf("\nQueue elements are:\n");
      for(i=front; i<=rear; i++)
      printf("%d\t",queue[i]);
   }
}
```

Figure 1 C program to do INSERT, DELETE and PRINT operations on queue

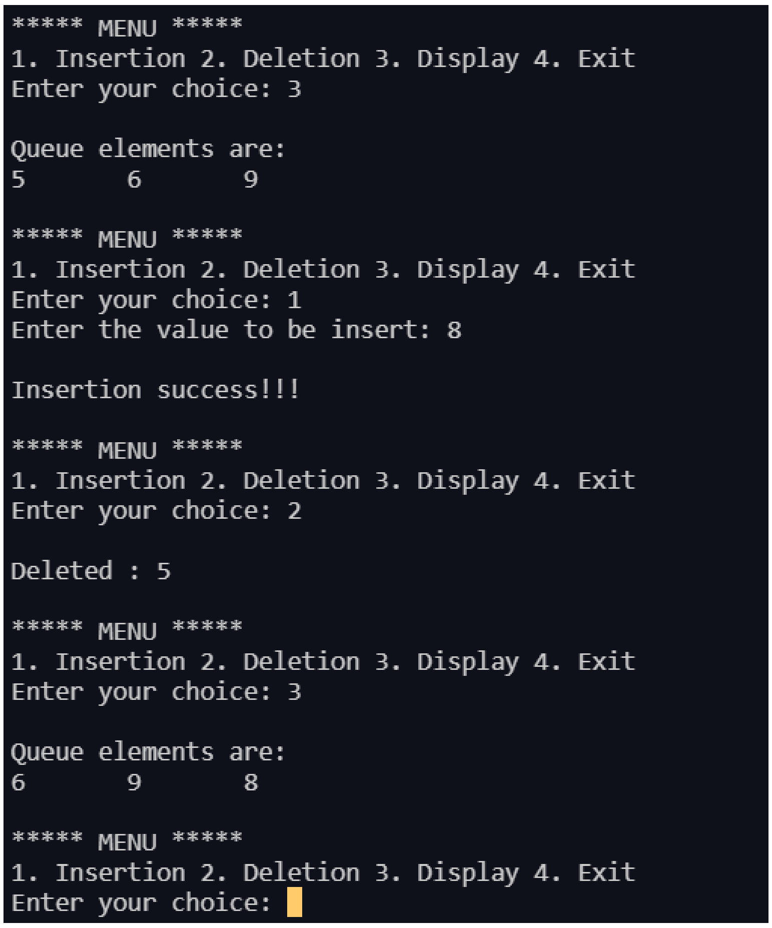
Name: SUBHENDU MAJI   Roll Number: 18ETCS002121



```c
//priority queue
#include <stdio.h>
#include <stdlib.h>

struct node{
  int val;
  int priority;
  struct node *next;
};

struct node *head;

struct node *newNode(int v,int p){
  struct node *temp = (struct node *)malloc(sizeof(struct node));
  temp->val = v;
  temp->priority = p;
  temp->next = NULL;
}

 void insert(int v,int p){
   struct node *start = head;
   struct node *temp = newNode(v,p);

   if (head->priority > p) {
     temp->next = head;
     head = temp;
   }

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

 void delete(){
   if (head==NULL){
     printf("Queue is empty\n");
   }
   else{
     struct node *temp = head;
     head = head->next;
     printf("deleted %d\n\n",temp->val);
     free(temp);
   }
 }

void print(){
  struct node *temp = head;
  while (temp!= NULL) {
    printf("%d -> ",temp->val);
    temp = temp->next;
  }
  printf("NULL\n");
}

 int main(int argc, char const *argv[]) {
   int c = 0;
   int v,p;
   printf("enter a value,priority for head:");
   scanf("%d%d",&v,&p);
   head = newNode(v,p);
   while (c!=4) {
     printf("\n1. Insert 2. Delete 3. Print 4. Exit\n");
     printf(" \t Enter your choice: ");
     scanf("%d",&c);
     if (c == 1) {
       printf("enter a value,priority: ");
       scanf("%d%d",&v,&p);
       insert(v,p);
     }
     else if (c == 2) {
       delete();
     }
     else if (c == 3) {
       print();
     }
     else if (c == 4) {
       printf("Exit");
     }
     else{
       printf("\ninvalid choice\n");
     }
   }
   return 0;
 }
```

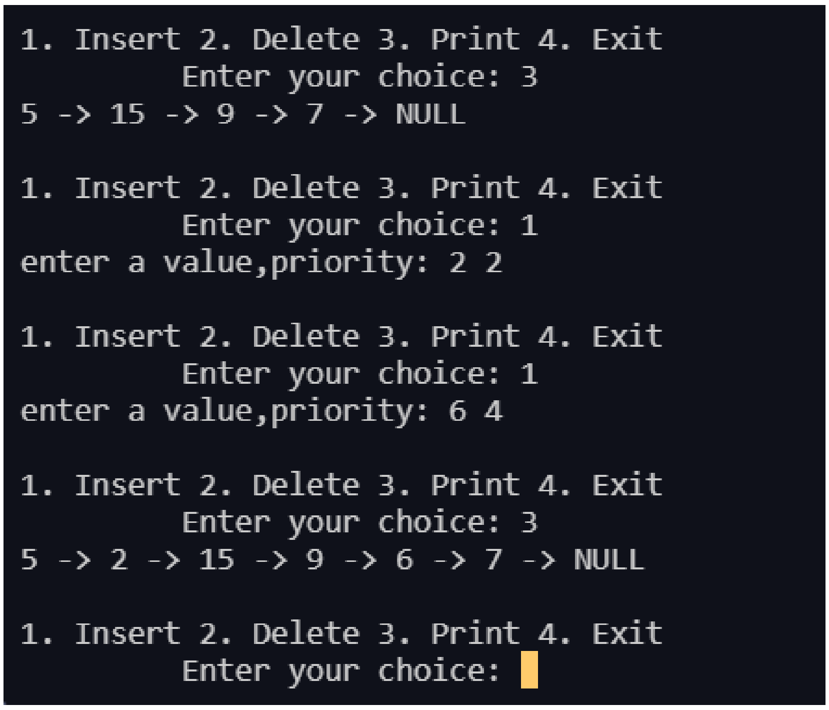
Figure 2 C program to implement priority queue using linked list

4. Presentation of Results



```
***** MENU *****
1. Insertion 2. Deletion 3. Display 4. Exit
Enter your choice: 3

Queue elements are:
5       6       9

***** MENU *****
1. Insertion 2. Deletion 3. Display 4. Exit
Enter your choice: 1
Enter the value to be insert: 8

Insertion success!!!

***** MENU *****
1. Insertion 2. Deletion 3. Display 4. Exit
Enter your choice: 2

Deleted : 5

***** MENU *****
1. Insertion 2. Deletion 3. Display 4. Exit
Enter your choice: 3

Queue elements are:
6       9       8

***** MENU *****
1. Insertion 2. Deletion 3. Display 4. Exit
Enter your choice:
```

Figure 3 output of program to INSERT, DELETE and PRINT operations on queue



```
1. Insert 2. Delete 3. Print 4. Exit
         Enter your choice: 3
5 -> 15 -> 9 -> 7 -> NULL

1. Insert 2. Delete 3. Print 4. Exit
         Enter your choice: 1
enter a value,priority: 2 2

1. Insert 2. Delete 3. Print 4. Exit
         Enter your choice: 1
enter a value,priority: 6 4

1. Insert 2. Delete 3. Print 4. Exit
         Enter your choice: 3
5 -> 2 -> 15 -> 9 -> 6 -> 7 -> NULL

1. Insert 2. Delete 3. Print 4. Exit
         Enter your choice:
```

Figure 4 output of program to implement priority queue using linked list

5. 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.