Documentation of Q1 (Explanations are written with the abbreviation "Exp:")

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


#define MAX 10
```

Exp: Libraries are called and a MAX is defined that is the maximum inputs the queue can take.

```
typedef struct node

{
        int priority;
        int time;

}NODE;
```

Exp: A struct is defined that will store the times and the respective priorities.

```
int insert(int data, int data_priority);

int delete();
int display();
int rear=-1;

int front=-1;
```

Exp: Functions are declared giving the variables rear and front a value of -1.

```
NODE queue[MAX];
```

queue is a variable of type struct is declared.

```
int main()
{
        int choice,data,data_priority, i;
        while(1)
                                                            // switch is used
to run through options
        {
                printf("1.Insert\n");
                printf("2.Delete\n");
                printf("3.Display\n");
                printf("4.Quit\n");
                printf("Enter your choice : ");
                scanf("%d", &choice);

                switch(choice)
                                                            // switches
between the main menu
                {
                 case 1:
                        insert(data, data_priority);
                        break;
                 case 2:
                        printf("Deleted time is %d\n",delete());

                        break;
                 case 3:
                        display();
                        break;
                 case 4:
                        exit(1);
                 default :
                        printf("Wrong choice\n");
                }

        }
}
```

Exp: In the main function a switch is used to switch between different options in the menu using the numbers.
pressing 1 takes us to insert function which allows us to insert the time and the priority of the process such the most eligible process is at the top.
pressing 2 takes us to delete function which deleted the process at the top since the most eligible process is at the top.
pressing 3 displays the input times and their priorities.
pressing 4 exits from the whole program.
default takes care such that only numbers from 1-4 are pressed and prints wrong choice for anything pressed except for the desired numbers.

```
int insert(int data, int data_priority)
```

```
{
        printf("Input the time taken :
");                     // data value and associated priority
is taken up
        scanf("%d",&data);
        printf("Enter its priority : ");
        scanf("%d",&data_priority);

    int i, j;

        if( rear== MAX-1 )
                                                            // to
check if the queue is full
        {
                printf("Queue Overflow\n");
                return 0;
        }

    if( front == -1 )

                front = 0;


        if((front==-1&& rear== -1) || data_priority <
queue[front].priority)   /*Lower priority value means higher
priority

                                                            if the
queue is empty or the priority associated with the inserted

                                                            data is
lower that the pre-existing data *If loop runs* */
        {

                for(i = rear ; i >= front ; i--)
        {
            queue[i + 1] = queue[i];
        }

         queue[front].time = data;
         queue[front].priority = data_priority;

         rear = rear + 1;
        }
        else
                                                     /* if the queue
is not empty and the inserted data has a priority value higher


than the previous numbers.    */
        {
        i = front;

                while(i <= rear)
        {
            if(data_priority <= queue[i].priority && data <=
queue[i].time)                    //lower priority element found
aswell as the element taking least time is also found
            {
                break;
            }
                        i++;
        }

        if(i > rear)
                                  //lower priority element not found
(insertion to be done at end)
        {
            rear = i;
        }
        else
                                          //lower priority element
found (insert before that element)
        {
            for(j = rear ; j >= i ; j--)
            {
                queue[j + 1] = queue[j];
            }

            rear++;
        }

        queue[i].time = data;
                         // data and priorities are stored
        queue[i].priority = data_priority;

         }

    display();
                                          // the inserted time and
priorities are seen after every insertion
    return 0;
}
```

Exp: In the insert function the value of time and the priority associated is taken from the user. If the rear is (MAX-1)
that means the queue is empty since the rear has already reached end of the queue. If the queue is empty (this is when the first element is being inserted to the queue)
or if the priority associated with the data is less that the priority associated with the pre existing process then the user's data and priority is inserted in the first place.
Basically the lowest priority is given more importance and get's the first priority.
Else if the queue is not empty but we have more than one process with same priority value then the execution time is checked to give priority. If two different execution times have

priority of "1" then process with less execution time is executed first, else the insertion takes place based on the priority associated with the process.

```
int delete()

{

        int data;
        if(front == -1 && rear== -1)

        {
                printf("Queue Underflow\n");
                exit(1);
        }
        else
        {
                data = queue[front].time;

        front++;
        }
        return data;
}
```

Exp: In delete function we check if the queue is empty, if it is not then process at the front is deleted.

```
int display()

{
        int  i = front;
        if( front== -1&& rear == -1 )
                printf("Queue is empty\n");
        else
        {       printf("Queue is :\n");

                printf("Time       Priority\n");
                while(i <= rear)
                {
                        printf("%5d        %5d\n", queue[i].time,
queue[i].priority);
                        i++;
                }
        }
        return 0;
}
```

Exp: To display the processes inserted in the queue first it i
checked f the queue is empty, if not the values are printed while
the
index of the processes is less that the value of rear.