```
struct queues
{
 int priority_start;
 int priority_end;
 int total_time = 0;
 process * p;
 ~~bool~~ bool executed = false;
 };
bool notcomplete (queues q[])
{
 bool a = false;
 int countInc = 0;
 for (int i = 0; i<3; i++)
  {
   countInc = 0;
   for (int j = 0 ; j < q[i].length; j++)
  {
   if (q[i].p[j].burst_time != 0)
   {
    a = true;
   }
   else
   {
    countInc += 1;
   }
 }
if (count Inc == a[i].length)
{
   a[i].executed = true;
  }
 }
 return a;
}
~~void sort_PS [queues a]~~
void sort_PS (queues a)
{
 for (int i = 1; i < q.length; i++)
  {
   if (q.p[J].Priority < q.p[J+1].priority)
   {
      process temp = q.p[J+1];
      q.p[J+1] = q.p[J];
       q.p[J] = temp;

     }
    }
   }
 }
void check complete Timer (queues a[])
{
 bool a = not complete (a);
 for (int i = 0; i < 3; i++)
  {
```

```
if (a[i].executed = =false)
{ for (int j=0; j< a[i].length; j++)
  {
    if (a[i].p[j].burst_time != 0)
        a[i].p[j].total_time+=1;
    }
  }
  a[i].total_time+=1;
 }
}
}

int main ()
{
a[0]. priority_start =7;
a[0]. priority_end = 9;
a[1]. priority _start = 4;
a[1]. priority _end = 6
a[2]. priority - start =1;
a[2]. priority - end = 3.

int no_of_processes, priority_of_process,
burst_time_of_process;
cout << "Enter the number of process\n"
cin >> no_of_processes;
```

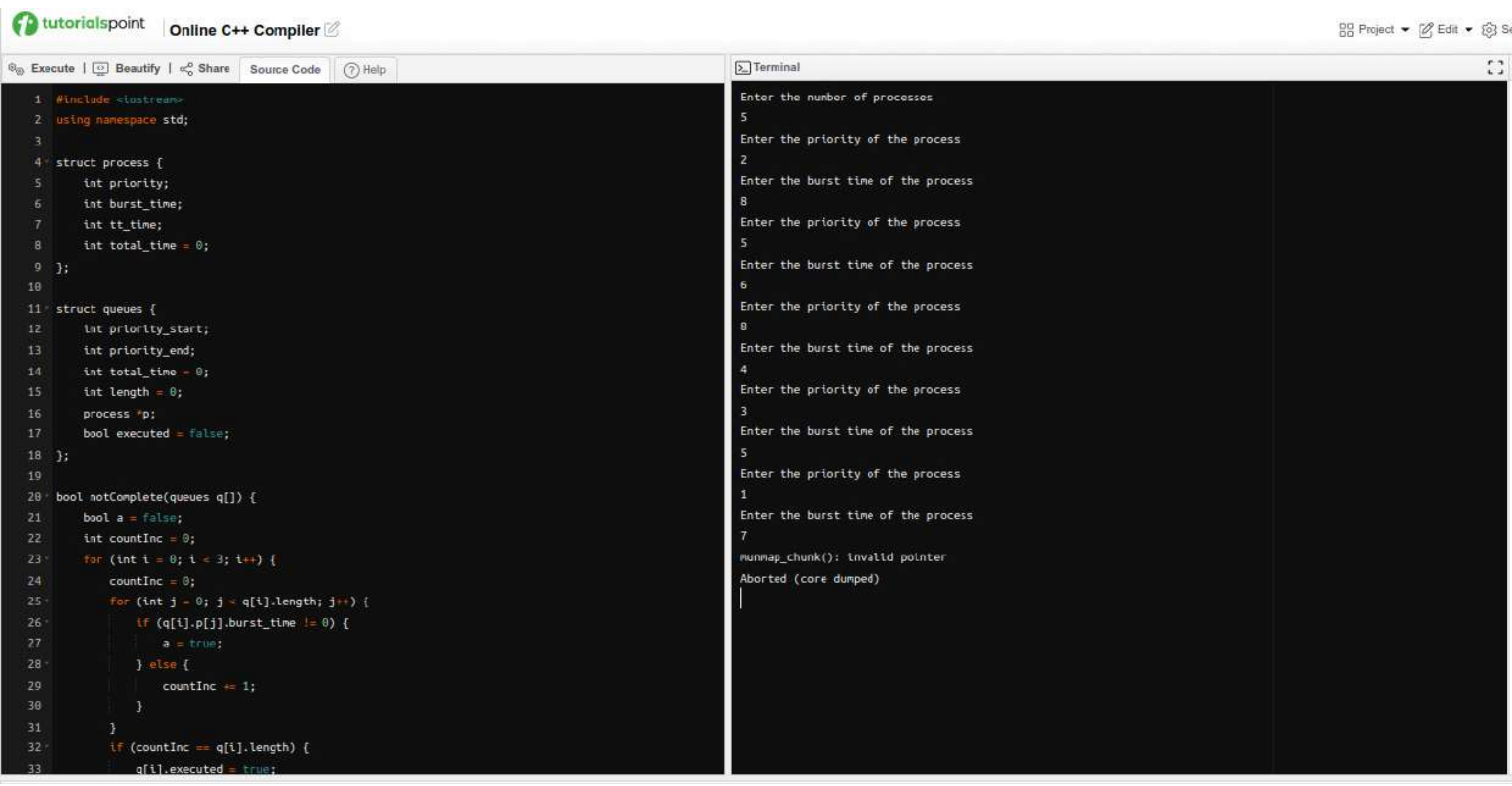
```
process P1[no_of_processes];
for (int i = 0; i < no_of_processes; i++)
{
  cout << "Enter the priority of the process\n";
  cin >> priority_of_process;
  cout << "Enter the burst time of the process\n";
  cin >> burst time_of_process;
  P1[i].priority = priority_of_process;
  P1[i].burst_time = burst_time_of_
                                process;
 for (int J = 0; J < 3; J++)
 {
 cout << "Enter the priority of the process\n";
cin >> priority_of_process;
cout << "Enter the burst time of the process
                                        \n";
cin >> burst_time_of_process;
~~P1[i]. priority = priority_of_process;~~
~~P1[i]. tt_time = burst_time_of_process;~~
 for (int J = 0; J < 3; J++)
 {
  if (v[J].priority_start <= priority_of_
                          process &&
          priority_of_process <= v[J].
   priority_end)
```

```
    {
        q[j]. length++;
      }
    }
  }
for (int i = 0; i< 3; i++)
{
  int len = q[i].length;
  q[i].p = new process[len];
}

   int a = 0;
   int b = 0;
   int c = 0;

for (int i = 0; i<3 ; i++)
{
for ( int j = 0 ; j < no_of_processes; j
{
if (( q[i]. priority_start <= p1[j].
                                  priority
                          priority_end))
{
   if (i==0)
   {
       q[i].p[a++] = p1[J];
   }
else if (i==1)
{
  q[i].p[b++] = p1[j];
  }
  else
  {
  q[i]. p[c++] =p1[j];
    }
   }
  }
 }
a-- ; b --; c--;
for(int =0 ; i<3; i++)
 {
 cout << " Queue" << i+1 << " : \t";
   for(int j = 0 ;   j < q[i].length; j++)
   {
    cout <<  q[i].p[j].priority << "->";
   }
   cout << " Null \n";
   }
         int timer =0;
          int 1 =-1;
            int rr_time =  4;
               int counter =0;
              int counter fs = 0;
```

Q. Out Put

```
Enter the number of process
5
Enter priority of the process
2
Enter the burst time of the process
8
Enter the priority of the process
5
Enter the burst time for the process
6
Enter the priority of the process
8
Enter the burst time for the process
4
Enter the priority of the process
3
Enter the burst time for process
5
Enter the priority of the process
1
Enter the burst time of of the process
7
```

6/10



```cpp
#include <iostream>
using namespace std;

struct process {
    int priority;
    int burst_time;
    int tt_time;
    int total_time = 0;
};

struct queues {
    int priority_start;
    int priority_end;
    int total_time = 0;
    int length = 0;
    process *p;
    bool executed = false;
};

bool notComplete(queues q[]) {
    bool a = false;
    int countInc = 0;
    for (int i = 0; i < 3; i++) {
        countInc = 0;
        for (int j = 0; j < q[i].length; j++) {
            if (q[i].p[j].burst_time != 0) {
                a = true;
            } else {
                countInc += 1;
            }
        }
        if (countInc == q[i].length) {
            q[i].executed = true;
```

```
Enter the number of processes
5
Enter the priority of the process
2
Enter the burst time of the process
8
Enter the priority of the process
5
Enter the burst time of the process
6
Enter the priority of the process
8
Enter the burst time of the process
4
Enter the priority of the process
3
Enter the burst time of the process
5
Enter the priority of the process
1
Enter the burst time of the process
7
munmap_chunk(): invalid pointer
Aborted (core dumped)
```