OS Lab 7 Assignment

Name = Poorab Gangwani

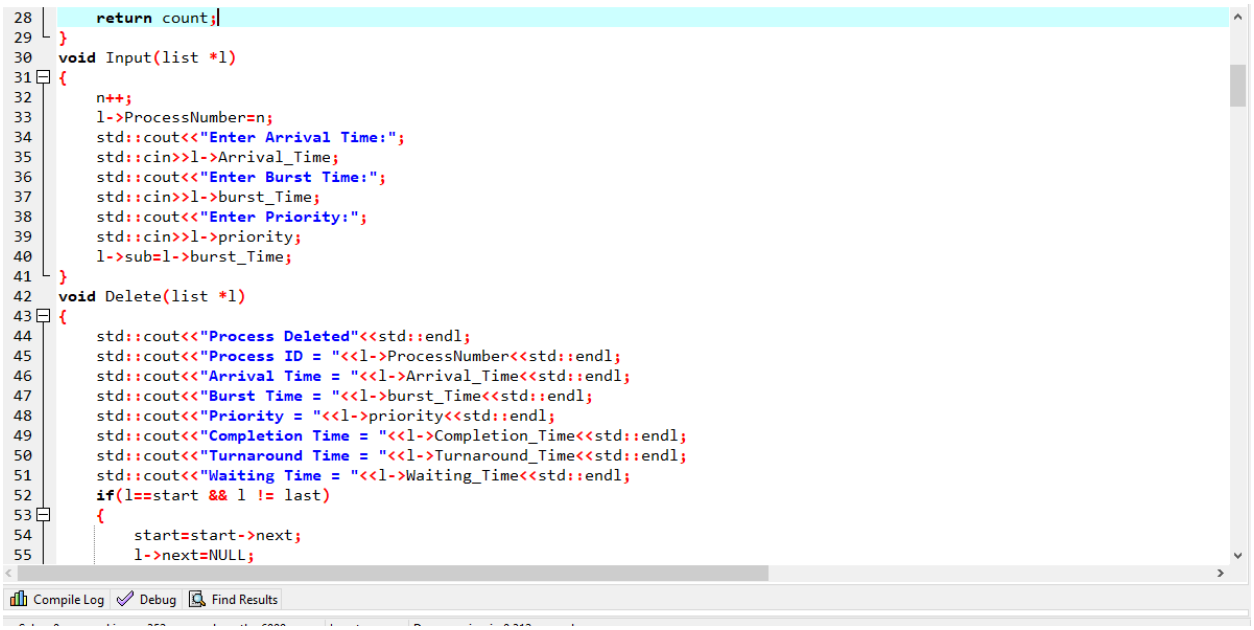
Class = 4B1

Roll-Number = CS191092

Code:

Assignment.cpp

```cpp
#include<iostream>
#include<math.h>
struct list{
    int ProcessNumber=0;
    int Arrival_Time=0;
    int burst_Time=0;
    int Turnaround_Time=0;
    int Completion_Time=0;
    int Waiting_Time=0;
    int priority=0;
    list *next=NULL;
    bool completed=false;
    int sub;
    bool In_Process=false;
    int dormant_time=0;
    bool repeat=false;
}*start=NULL,*last=NULL;
int n=0;
int Total_Processes()
{
    list *temp=start;
    int count=0;
    while(temp != NULL)
    {
        count++;
        temp=temp->next;
    }
    return count;
```



```
28      return count;
29  }
30  void Input(list *l)
31  {
32      n++;
33      l->ProcessNumber=n;
34      std::cout<<"Enter Arrival Time:";
35      std::cin>>l->Arrival_Time;
36      std::cout<<"Enter Burst Time:";
37      std::cin>>l->burst_Time;
38      std::cout<<"Enter Priority:";
39      std::cin>>l->priority;
40      l->sub=l->burst_Time;
41  }
42  void Delete(list *l)
43  {
44      std::cout<<"Process Deleted"<<std::endl;
45      std::cout<<"Process ID = "<<l->ProcessNumber<<std::endl;
46      std::cout<<"Arrival Time = "<<l->Arrival_Time<<std::endl;
47      std::cout<<"Burst Time = "<<l->burst_Time<<std::endl;
48      std::cout<<"Priority = "<<l->priority<<std::endl;
49      std::cout<<"Completion Time = "<<l->Completion_Time<<std::endl;
50      std::cout<<"Turnaround Time = "<<l->Turnaround_Time<<std::endl;
51      std::cout<<"Waiting Time = "<<l->Waiting_Time<<std::endl;
52      if(l==start && l != last)
53      {
54          start=start->next;
55          l->next=NULL;
```

Compile Log Debug Find Results

Assignment.cpp

```
55          l->next=NULL;
56          delete l;
57      }
58      else if(l==last && last==start)
59      {
60          start=NULL;
61          last=NULL;
62      }
63      else if(l==last && last != start)
64      {
65          list *temp=start;
66          while(temp->next != last)
67          {
68              temp=temp->next;
69          }
70          list *temp2=last;
71          last=temp;
72          temp->next=NULL;
73          delete temp2;
74      }
75      else
76      {
77          list *temp=start;
78          while(temp->next != l)
79          {
80              temp=temp->next;
81          }
82          temp->next=l->next;
```

Assignment.cpp

```
82          temp->next=l->next;
83          l->next=NULL;
84          delete l;
85      }
86  }
87  void Table()
88  {
89      std::cout<<"Process ID"<<"\t"<<"Arrival Time"<<"\t"<<"Burst Time"<<"\t"<<"Priority"<<"\t"<<"Completion Time"<<"\t"<<"Turr
90      for(list *z=start;z != NULL;z=z->next)
91      {
92          std::cout<<z->ProcessNumber<<"\t\t"<<z->Arrival_Time<<"\t\t"<<z->burst_Time<<"\t\t"<<z->priority<<"\t\t"<<z->Completi
93      }
94  }
95  list* Previous(list *l)
96  {
97      list *temp=start;
98      while(temp->next != l)
99      {
100         temp=temp->next;
101     }
102     return temp;
103 }
104 void Discard(list *l)
105 {
106     list *temp=start;
107     if(l==start)
108     {
109         start=start->next;
```

Assignment.cpp

```
109         start=start->next;
110         l->next=NULL;
111     }
112     else
113     {
114         list *x=Previous(l);
115         if(l==last)
116         {
117             x->next=NULL;
118             last=x;
119         }
120         else
121         {
122             x->next=l->next;
123             l->next=NULL;
124         }
125     }
126 }
127 int Priority()
128 {
129     int i=0;
130     for(list *l=start;l != NULL;l=l->next)
131     {
132         if(l->priority>i)
133         {
134             i=l->priority;
135         }
136     }
```

Assignment.cpp

```
136     }
137     return i+1;
138 }
139 void Display(list *z)
140 {
141     std::cout<<"Process ID"<<"\t"<<"Arrival Time"<<"\t"<<"Burst Time"<<"\t"<<"Priority"<<"\t"<<"Completion Time"<<"\t"<<"Turr
142     std::cout<<z->ProcessNumber<<"\t\t"<<z->Arrival_Time<<"\t\t"<<z->burst_Time<<"\t\t"<<z->priority<<"\t\t"<<z->Completion_T
143 }
144 void swap(int &x,int &y)
145 {
146     int temp=x;
147     x=y;
148     y=temp;
149 }
150 void Priority_Insertion()
151 {
152     list *l=new list;
153     Input(l);
154     if(start==NULL)
155     {
156         start=l;
157         last=l;
158     }
159     else if(l->priority < start->priority)
160     {
161         l->next=start;
162         start=l;
163     }
```

Assignment.cpp

```
163     }
164     else if(l->priority >last->priority)
165     {
166         last->next=l;
167         last=l;
168     }
169     else if(l->priority==start->priority && start->Arrival_Time>l->Arrival_Time)
170     {
171         l->next=start;
172         start=l;
173     }
174     else if(l->priority==start->priority && start->Arrival_Time<l->Arrival_Time)
175     {
176         l->next=start->next;
177         start->next=l;
178     }
179     else if(l->priority==last->priority && last->Arrival_Time<l->Arrival_Time)
180     {
181         last->next=l;
182         last=l;
183     }
184     else
185     {
186         list *temp=start,*z=last;
187         while(1)
188         {
189         if(z->priority > l->priority && z != start)
190         {
```

Assignment.cpp

```
190         {
191             z=Previous(z);
192         }
193         else if(z->priority > l->priority && z==start)
194         {
195             l->next=start;
196             start=l;
197             break;
198         }
199         else if(z->priority < l->priority && z==last)
200         {
201             last->next=l;
202             last=l;
203             break;
204         }
205         else if(z->priority==l->priority)
206         {
207             if(z->Arrival_Time > l->Arrival_Time)
208             {
209                 list *x=Previous(z);
210                 x->next=l;
211                 l->next=z;
212                 break;
213             }
214             else if(z->Arrival_Time < l->Arrival_Time)
215             {
216                 l->next=z->next;
217                 z->next=l;
```

Assignment.cpp

```
217                 z->next=l;
218                 break;
219             }
220         }
221         else if(z->priority < l->priority)
222         {
223             l->next=z->next;
224             z->next=l;
225             break;
226         }
227         }
228     }
229 }
230 void Aging(list *l)
231 {
232     l->priority-=1;
233     list *z=last;
234     while(1)
235     {
236         if(z->priority > l->priority && z != start)
237         {
238             z=Previous(z);
239         }
240         else if(z->priority > l->priority && z==start)
241         {
242             l->next=start;
243             start=l;
244             break;
```

Assignment.cpp

```
244             break;
245         }
246         else if(z->priority < l->priority && z==last)
247         {
248             last->next=l;
249             last=l;
250             break;
251         }
252         else if(z->priority==l->priority)
253         {
254             if(z->Arrival_Time > l->Arrival_Time && z==start)
255             {
256                 l->next=start;
257                 start=l;
258             }
259             else if(z->Arrival_Time > l->Arrival_Time)
260             {
261                 list *x=Previous(z);
262                 x->next=l;
263                 l->next=z;
264                 break;
265             }
266             else if(z->Arrival_Time < l->Arrival_Time)
267             {
268                 l->next=z->next;
269                 z->next=l;
270                 break;
271             }
```

Assignment.cpp

```
271             }
272         }
273         else if(z->priority < l->priority)
274         {
275             l->next=z->next;
276             z->next=l;
277             break;
278         }
279     }
280 }
281 int main()
282 {
283     int size,time=0;
284     list *temp=NULL;
285     std::cout<<"Enter Initial Number of Processes:";
286     std::cin>>size;
287     for(int i=0;i<size;i++)
288     {
289         Priority_Insertion();
290     }
291     while(start != NULL && last != NULL)
292     {
293         int largest_Priority=Priority();
294         for(list *l=start;l != NULL;l=l->next)
295         {
296             if(l->Arrival_Time<=time && l->sub>0 && l->priority<largest_Priority)
297             {
298                 largest_Priority=l->priority;
```

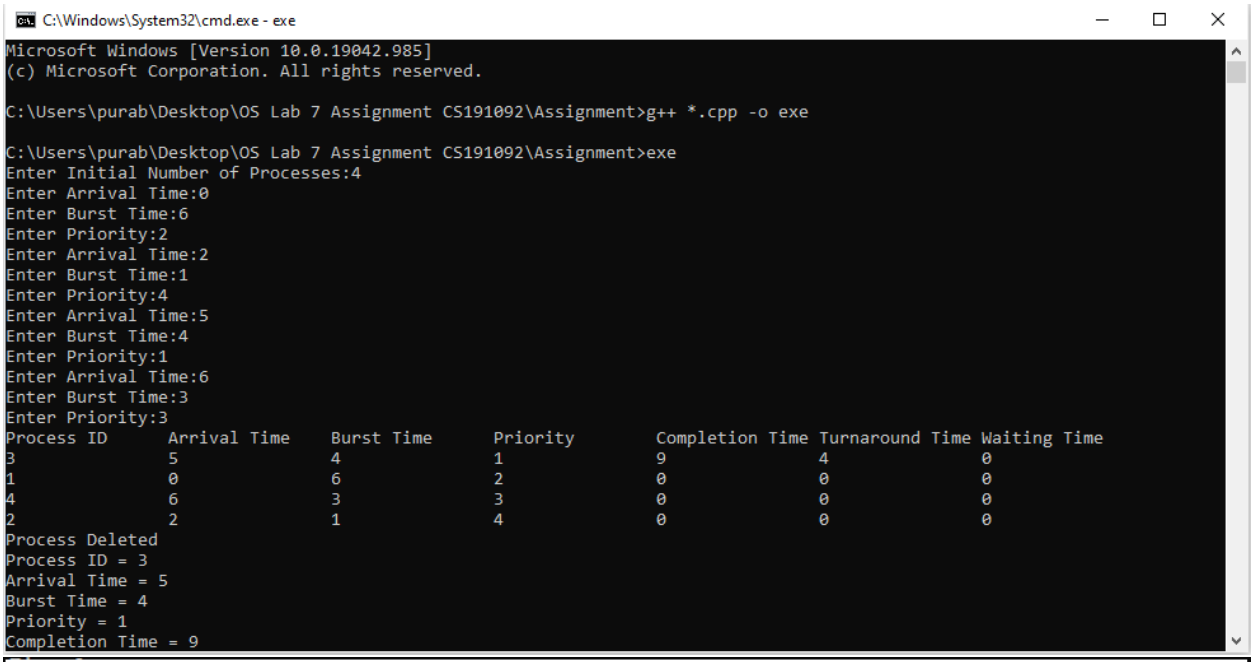
Assignment.cpp

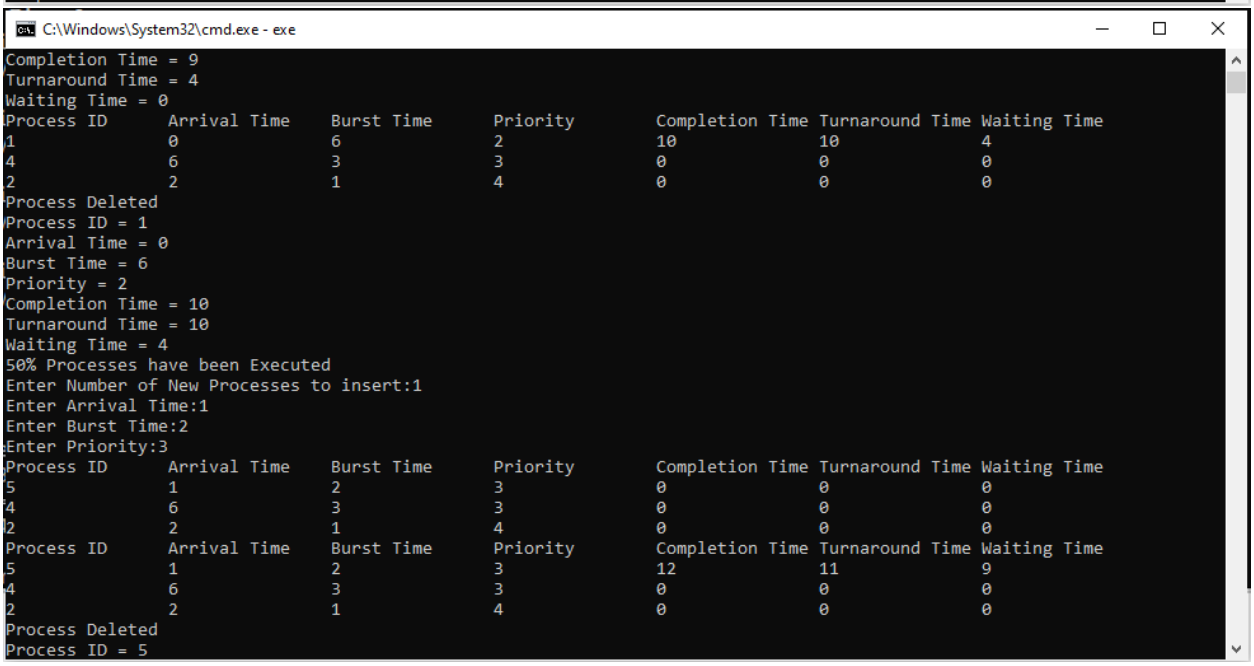
```
298                 largest_Priority=l->priority;
299                 temp=l;
300                 l->In_Process=true;
301             }
302         }
303         temp->sub--;
304         time++;
305         for(list *l=start;l != NULL;l=l->next)
306         {
307             if(l->Arrival_Time < time && l->In_Process==false)
308             {
309                 l->dormant_time+=1;
310             }
311         }
312         if(temp->sub==0)
313         {
314             temp->completed=true;
315             temp->Completion_Time=time;
316             temp->Turnaround_Time=(temp->Completion_Time)-(temp->Arrival_Time);
317             temp->Waiting_Time=(temp->Turnaround_Time)-(temp->burst_Time);
318             if(temp->Waiting_Time<0)
319             {
320                 temp->Waiting_Time=0;
321             }
322             Table();
323             Delete(temp);
324             if(size%2==0 && Total_Processes()==(size/2) || size%2 != 0 && Total_Processes()==floor(size/2))
325             {
```

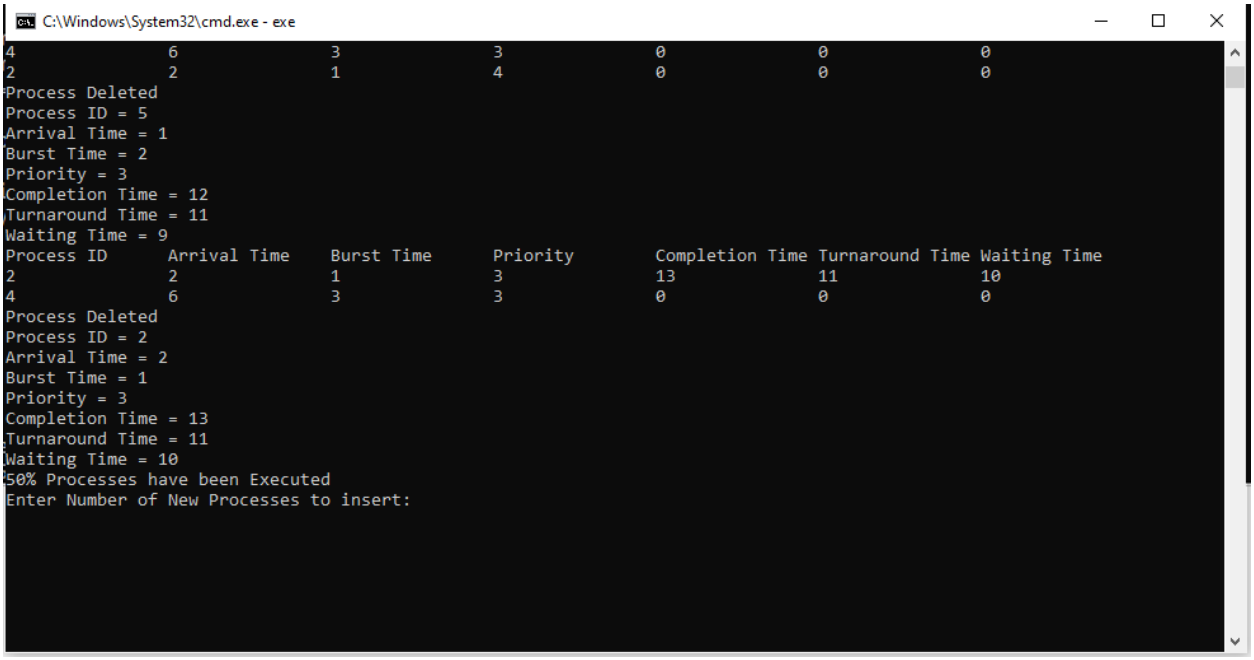
Assignment.cpp

```
325             {
326                 int n;
327                 std::cout<<"50% Processes have been Executed\n";
328                 std::cout<<"Enter Number of New Processes to insert:";
329                 std::cin>>n;
330                 size=n+Total_Processes();
331                 for(int i=0;i<n;i++)
332                 {
333                     Priority_Insertion();
334                 }
335                 Table();
336             }
337         }
338         for(list *l=start;l != NULL;l=l->next)
339         {
340             for(list *z=start;z != NULL;z=z->next)
341             {
342                 if(z->dormant_time==10)
343                 {
344                     z->dormant_time=0;
345                     Discard(z);
346                     Aging(z);
347                     break;
348                 }
349             }
350         }
351     }
352 }
```

Output:


C:\Windows\System32\cmd.exe - exe
Microsoft Windows [Version 10.0.19042.985]
(c) Microsoft Corporation. All rights reserved.
C:\Users\purab\Desktop\OS Lab 7 Assignment CS191092\Assignment>g++ *.cpp -o exe
C:\Users\purab\Desktop\OS Lab 7 Assignment CS191092\Assignment>exe
Enter Initial Number of Processes:4
Enter Arrival Time:0
Enter Burst Time:6
Enter Priority:2
Enter Arrival Time:2
Enter Burst Time:1
Enter Priority:4
Enter Arrival Time:5
Enter Burst Time:4
Enter Priority:1
Enter Arrival Time:6
Enter Burst Time:3
Enter Priority:3
Process ID Arrival Time Burst Time Priority Completion Time Turnaround Time Waiting Time
3 5 4 1 9 4 0
1 0 6 2 0 0 0
4 6 3 3 0 0 0
2 2 1 4 0 0 0
Process Deleted
Process ID = 3
Arrival Time = 5
Burst Time = 4
Priority = 1
Completion Time = 9


C:\Windows\System32\cmd.exe - exe
Completion Time = 9
Turnaround Time = 4
Waiting Time = 0
Process ID Arrival Time Burst Time Priority Completion Time Turnaround Time Waiting Time
1 0 6 2 10 10 4
4 6 3 3 0 0 0
2 2 1 4 0 0 0
Process Deleted
Process ID = 1
Arrival Time = 0
Burst Time = 6
Priority = 2
Completion Time = 10
Turnaround Time = 10
Waiting Time = 4
50% Processes have been Executed
Enter Number of New Processes to insert:1
Enter Arrival Time:1
Enter Burst Time:2
Enter Priority:3
Process ID Arrival Time Burst Time Priority Completion Time Turnaround Time Waiting Time
5 1 2 3 0 0 0
4 6 3 3 0 0 0
2 2 1 4 0 0 0
Process ID Arrival Time Burst Time Priority Completion Time Turnaround Time Waiting Time
5 1 2 3 12 11 9
4 6 3 3 0 0 0
2 2 1 4 0 0 0
Process Deleted
Process ID = 5


C:\Windows\System32\cmd.exe - exe
4 6 3 3 0 0 0
2 2 1 4 0 0 0
Process Deleted
Process ID = 5
Arrival Time = 1
Burst Time = 2
Priority = 3
Completion Time = 12
Turnaround Time = 11
Waiting Time = 9
Process ID Arrival Time Burst Time Priority Completion Time Turnaround Time Waiting Time
2 2 1 3 13 11 10
4 6 3 3 0 0 0
Process Deleted
Process ID = 2
Arrival Time = 2
Burst Time = 1
Priority = 3
Completion Time = 13
Turnaround Time = 11
Waiting Time = 10
50% Processes have been Executed
Enter Number of New Processes to insert: