Practical No 7

Aim : Develop, debug and execute a C program to simulate the priority CPU scheduling algorithms to find turnaround time and waiting time.

Apparatus: Computer system with windows installed in it.
Mingw compiler for C/C++, and a text editor for developing C code file (Dev C++).

Theory :

What is Priority scheduling?

- Priority scheduling is a scheduling algorithm which decides.
- The Shortest Job First algorithm and the Shortest remaining time first algorithm can be said to be a general case example of priority scheduling such that the shorter time is given higher priority and the algorithms with longer time are given the less priority.
- A priority is associated with a process by the CPU or the user and the priority is saved in the processes.
- If two or more than two processes with equal priority arrive in the ready queue, then these processes are scheduled in First-come, First-serve order.
- Priority can be defined either internal or external order

What happens in Preemptive and nonpreemptive priority scheduling?

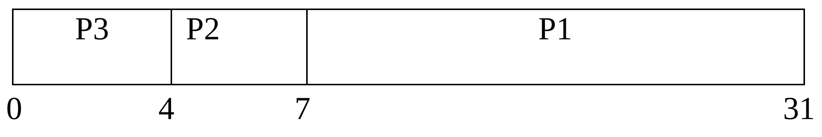
- In preemptive priority scheduling what happens is that the process with the highest priority is executed first, and if a process arrives with a higher priority, the current process is preempted, context switch occurs, and the process with higher priority is provided the CPU and the resources and it starts its execution.
- In nonpreemptive priority scheduling algorithm, if a process with a higher priority arrives in the ready queue, the process which is currently executing will not be affected, the new process with the highest priority will have to wait until the current process gets executed.
- Once the current process is executed, the process with the highest priority is selected by the CPU scheduler/ short term scheduler and the resources are allocated to it and it will start its execution.

What is a Starvation and what is Aging?

- Starvation
  - In priority scheduling, if a process of less priority is submitted for execution, there is a possibility that the system might be heavily loaded and crammed up with high priority processes.
  - This might result in low priority processes not getting CPU time and not being executed at all, this phenomenon of getting stuck due to low priority is called as starvation.
  - A classic example is that, when they shut down the IBM 9074 at MIT in the year 1973, they found a low priority process which never got executed since its submission in the year 1967.
- Aging
  - Aging is the solution to the problem of starvation.
  - The idea behind aging is to increase the priority of a process gradually with time.
  - Example, if in a computer system there is an option of setting a priority between 0 to 127, and the priority of a process is set to the lowest, i.e. 127, with aging if we set the processes which are not executed to increase priority after every 15 minute by 1, then the process with 127 priority will eventually attain higher priority until it gets its priority to 0 and it will then get executed eventually.

<u>Example</u> :

Process	**Burst Time**	**Priority**
P1	24	3
P2	3	2
P3	4	1



Waiting time for p1 = 7

Waiting time for p2 = 4

Waiting time for p3 = 0

Average waiting time =p1 + p2 + p3 / 3

= (7 + 4 + 0)/ 3

= 11/3

= 3.67ms

Turnaround time for p1 = 31ms

Turnaround time for p2 = 7ms

Turnaround time for p3 = 4ms

Average turnaround time = 31 + 7 + 4 / 3

= 42 / 3

= 14ms

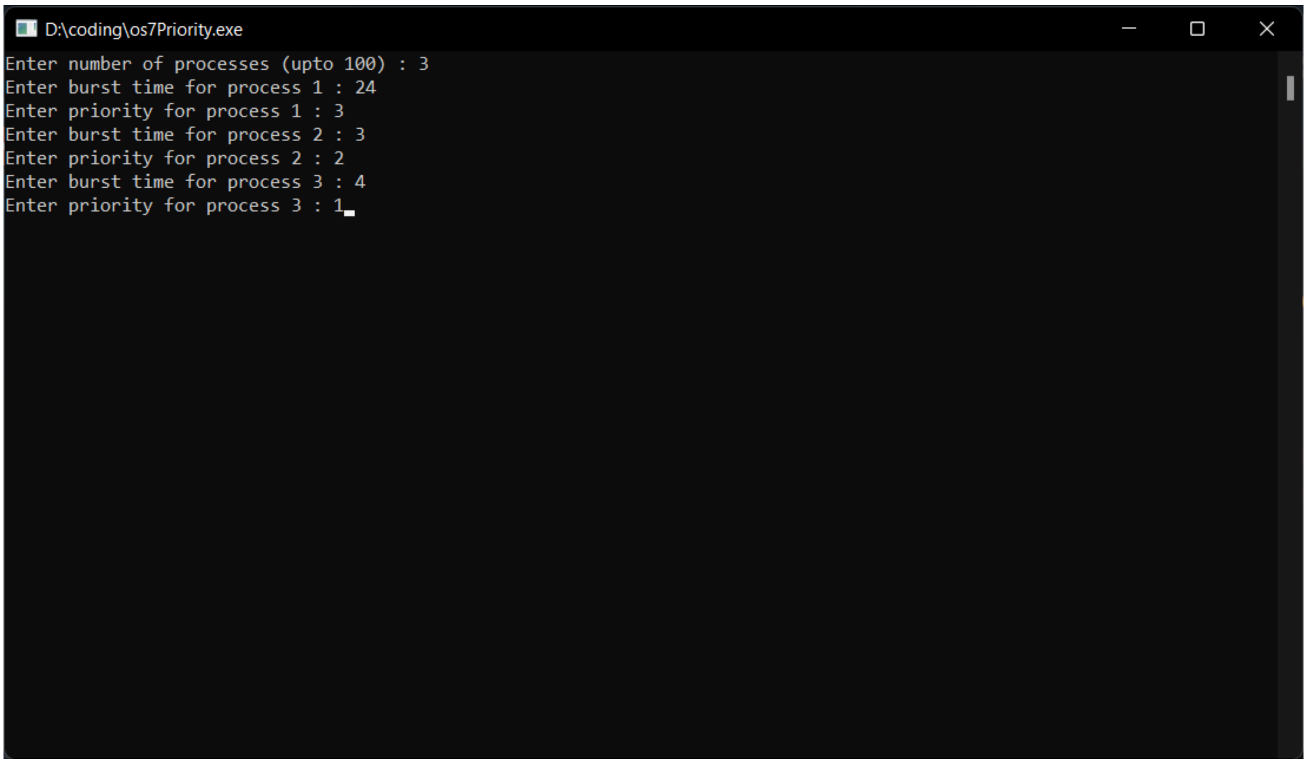
Code:

```
#include<stdio.h>
#define true 1
#define false 0


struct variables{
  int n;
  int priority[100];
  int burst_time[100];
  int turn_around_time[100];
  int waiting_time[100];

  struct Total{
    float waiting_time;
    float turn_around_time;
  }total;

  struct Average{
    float waiting_time;
    float turn_around_time;
  }average;

}data;

void getValues(){
  for(int i = 0; i < data.n; i++){
    printf("Enter burst time for process %d : ",i+1);
    scanf("%d",&data.burst_time[i]);
```

```c
    printf("Enter priority for process %d : ",i+1);
    scanf("%d",&data.priority[i]);
  }
}

void priorityScheduling(){
  int time = 0;
  int priority[data.n];

  for(int i = 0; i <data.n; i++)
    priority[i] = data.priority[i];

  int arr[data.n];

    int highest_priority = 99999;
    for(int i = 0; i < data.n; i++){
      int high = 100000;
      int index;
      for(int j = 0; j < data.n; j++){
        if(priority[j] < high){
          high = priority[j];
          index = j;
        }
      }
      arr[i] = index;
      priority[arr[i]] = 100001;
    }


    data.total.turn_around_time = 0;
    data.total.waiting_time = 0;

    for(int i = 0; i < data.n; i++){
      int n = arr[i];
      time += data.burst_time[n];
      data.waiting_time[n] = time - data.burst_time[n];
      data.turn_around_time[n] = time;
      data.total.turn_around_time +=  data.turn_around_time[n];
      data.total.waiting_time += data.waiting_time[n];
    }

    data.average.turn_around_time = data.total.turn_around_time / (float) data.n;
    data.average.waiting_time = data.total.waiting_time / (float) data.n;

}
```

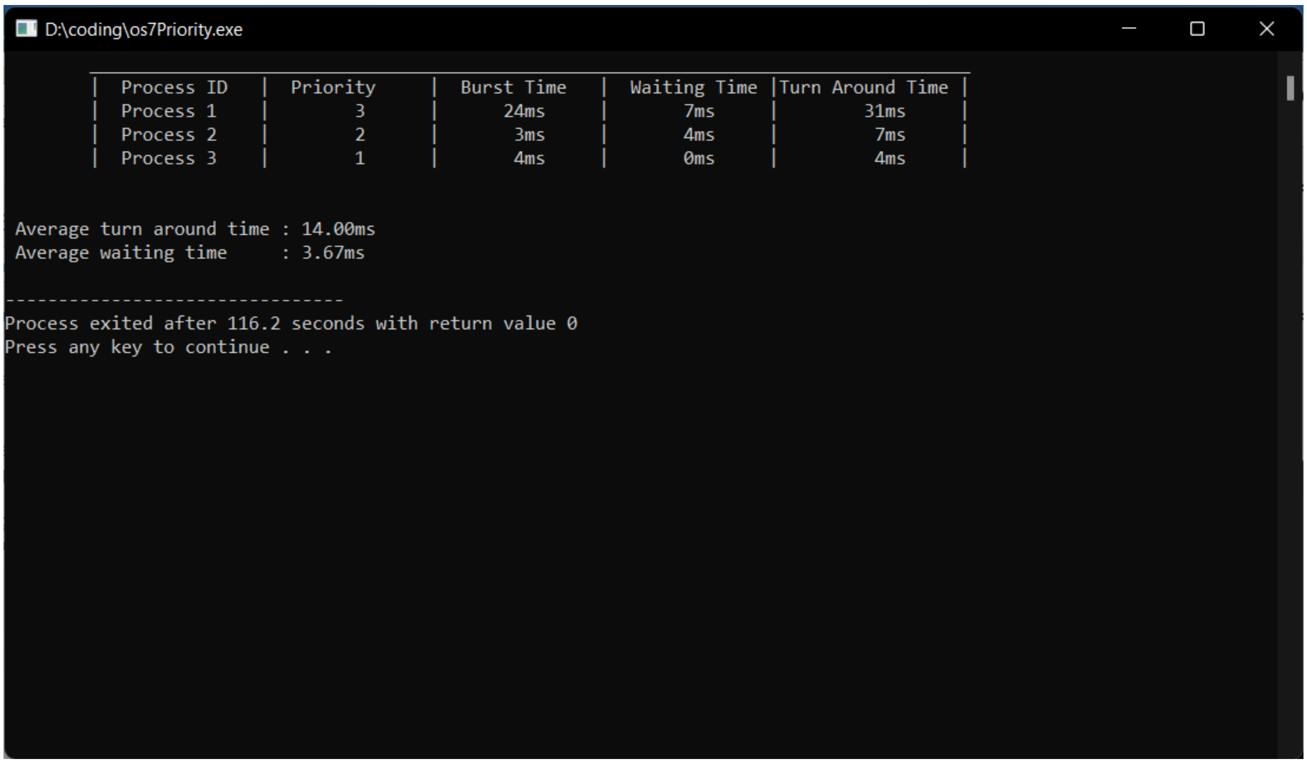
```c
void println(char ch){
  for(int i = 0; i < 83 ; i++)
    printf("%c",ch);
  printf("\n");
}

void display(){

  system("cls");
  printf("\t");
  println('_');
  printf("\t|  Process ID\t|  Priority\t|  Burst Time\t|  Waiting Time\t|Turn Around
Time |\n");
  for(int i = 0; i < data.n; i++ ){
    printf("\t|  Process %d\t| %8d\t|%8dms\t|%8dms\t|%10dms\t
|\n",i+1,data.priority[i],data.burst_time[i],data.waiting_time[i],data.turn_around_t
ime[i]);
  }


  printf("\n\n Average turn around time : %.2fms\n",data.average.turn_around_time);
  printf(" Average waiting time     : %.2fms\n",data.average.waiting_time);
}

int main(){
  printf("Enter number of processes (upto 100) : ");
  scanf("%d",&data.n);
  getValues();
  priorityScheduling();
  display();

}
```

Output:


D:\coding\os7Priority.exe

```
Enter number of processes (upto 100) : 3
Enter burst time for process 1 : 24
Enter priority for process 1 : 3
Enter burst time for process 2 : 3
Enter priority for process 2 : 2
Enter burst time for process 3 : 4
Enter priority for process 3 : 1
```

7.1. Inputting Data


D:\coding\os7Priority.exe

```
        ___________________________________________________________________________________
        |  Process ID   |  Priority       |  Burst Time    |  Waiting Time  |Turn Around Time |
        |  Process 1    |         3       |     24ms       |      7ms       |      31ms       |
        |  Process 2    |         2       |      3ms       |      4ms       |       7ms       |
        |  Process 3    |         1       |      4ms       |      0ms       |       4ms       |


 Average turn around time : 14.00ms
 Average waiting time     : 3.67ms

--------------------------------
Process exited after 116.2 seconds with return value 0
Press any key to continue . . .
```

7.2. Outputting Data

Conclusion:

Hence, by performing this practical I got to know about the concept of Priority scheduling. I also developed, debugged and executed a c program to simulate the Priority scheduling algorithm (Nonpreemptive) and found out the turnaround time and the waiting time using the program.