Problem Statement:

1. a. Design a CPU scheduler for jobs whose execution profiles will be in a file that is to be read and appropriate scheduling algorithm to be chosen by the scheduler. Format of the profile: -1 (Each information is separated by blank space and each job profile ends with -1. Lesser priority number denotes higher priority process with priority number 1 being the process with highest priority.) Example: 2 3 4 100 2 200 3 25 -1 7 1 8 60 10 ….. -1 etc. Testing: a. Create job profiles for 30 jobs and use three different scheduling algorithms (FCFS, Priority and Round Robin (time slice: 25) b. Compare the average waiting time, turnaround time of each process for the different scheduling algorithms.

```
#include<bits/stdc++.h>
using namespace std;
#define TIMEQUANTUM 25
//creating a Process class for uniquely representing each process and its
attributes
class Process
{
   public:
   int choice;
   int job_id,priority,arrival_time;
   queue<int> burst_time,io_time;
   vector<int> burst_time_arr,io_time_arr; //store these to print the chart
required for every job
   int current_time_io;//To track the current time of I/O process for each of
the process
   int total_burst_time;
    Process(int id,int p=0,int ch=1)
    {
       job_id=id;
       total_burst_time=0;
       choice=ch;
       if (choice==2)
       priority=p;
    }
    bool operator< (const Process &p) const{
        if (choice==1)
        return current_time_io>p.current_time_io;
        else if(choice==2)
        return priority<p.priority;
        else return current_time_io>p.current_time_io;
    }

};
void Read_Process_Data(string filename,queue<pair<Process,int> >
&Arrival_Time,int &n,unordered_map<int ,int >
&Arrival_Time_Map,unordered_map<int,int> &mp)
{
   ifstream file;
   file.open(filename);
   string current_word;
```

```
   vector<int> v;

   while (file >> current_word)
   {
        if(current_word=="-1"){
            n++;//tracks the number of processes read till now
            int id=v[0];
            Process p(id);
            bool flag=true;
            for(int i=3;i<v.size();i++){
                if(flag)
                {
                    p.burst_time.push(v[i]);//store in the cpu-burst queue
                    p.burst_time_arr.push_back(v[i]);
                    p.total_burst_time+=v[i];// add the total burst time for
the process
                }
                else
                {
                    p.io_time.push(v[i]); //else push to the i/o queue
                    p.io_time_arr.push_back(v[i]);
                }
                flag=!flag;
            }
            Arrival_Time.push({p,v[2]});//pair (process p,arrivaltime for
process p)
            Arrival_Time_Map[p.job_id]=v[2]; //store the arrival time for the
process p
            mp[p.job_id]=p.total_burst_time; // store the total burst time for
process p
            v.clear();
        }else
            v.push_back(stoi(current_word));
    }
    file.close(); //reading over , close the file now

}
void First_Come_First_Serve(string filename)
{
    //defining the necessary data structures for the algorithm
```

```
    Process current_process(-1);
    vector<Process> print_details;
    queue<pair<Process,int> > Process_Arrival_Queue;
    unordered_map<int ,int > Completion_Time;
    unordered_map<int ,int > Arrival_Time_Map;
    unordered_map<int,int> mp;
    priority_queue<Process> Input_Output;
    queue<Process> ready_queue;
    int n=0;

    Read_Process_Data(filename,Process_Arrival_Queue,n,Arrival_Time_Map,mp);


    for(int current_time=0;;){

        while(!Process_Arrival_Queue.empty() &&
Process_Arrival_Queue.front().second<=current_time){
            ready_queue.push(Process_Arrival_Queue.front().first);
            Process_Arrival_Queue.pop();
        }

        while(!Input_Output.empty() &&
Input_Output.top().current_time_io<=current_time){
            if(Input_Output.top().burst_time.size()==0){
                Completion_Time[Input_Output.top().job_id]=current_time;
            }else
                ready_queue.push(Input_Output.top());
            Input_Output.pop();
        }

        if(!ready_queue.empty()){
            current_process=ready_queue.front();
            ready_queue.pop();
            current_time+=current_process.burst_time.front();
            current_process.burst_time.pop();
            if(current_process.io_time.size()==0){
                Completion_Time[current_process.job_id]=current_time;
            }else if(current_process.io_time.size()){
                current_process.current_time_io=current_time+current_process.i
o_time.front();
```

```
                current_process.io_time.pop();
                Input_Output.push(current_process);
            }
        }
        else
            current_time++;

        if(ready_queue.size()==0 && Input_Output.size()==0){
            break;
        }
    }

    float avg_turnaround_time=0,avg_waiting_time=0;

    for(auto itr:Completion_Time)
    {
       cout<<"\n";
       int value1=itr.second-Arrival_Time_Map[itr.first];
       int value2=value1-mp[itr.first];
       cout<<"JobId:"<<itr.first<<"\nTurnaround Time:"<<value1<<"\nWaiting
Time:"<<value2<<"\n";
       avg_turnaround_time+=value1;
       avg_waiting_time+=value2;
       cout<<"\n";
    }

    cout<<"The average turnaround time is :"<<avg_turnaround_time/n<<"\n";
    cout<<"The average waiting time is :"<<avg_waiting_time/n<<"\n";

}
void Priority_Scheduling(string filename)
{
    Process current_process(-1,-1, 2);//passing the choice 2 for the
overloading of the boolean operator

    queue<pair<Process,int>> Process_Arrival_Queue;
    unordered_map<int,int> Completion_Time;
    vector<Process> Input_Output;
    priority_queue<Process> ready_queue;//priority queue instead of normal
queue
```

```
    unordered_map<int ,int > Arrival_Time_Map;
    unordered_map<int,int> mp;

    int n=0;

    Read_Process_Data(filename,Process_Arrival_Queue,n,Arrival_Time_Map,mp);

    for(int current_time=0;;){
        vector<Process> temp;
        for(int j=0;j<Input_Output.size();j++){
            if(Input_Output[j].current_time_io<current_time){
                if(Input_Output[j].burst_time.size()==0)
                    Completion_Time[Input_Output[j].job_id]=current_time;
                else
                    ready_queue.push(Input_Output[j]);
            }else
                temp.push_back(Input_Output[j]);
        }
        Input_Output=temp;
        if(ready_queue.size()){
            current_process=ready_queue.top();
            ready_queue.pop();
            current_process.burst_time.front()-=1;
            if(current_process.burst_time.front()==0){
                current_process.burst_time.pop();
                if(current_process.io_time.size()==0)Completion_Time[current_p
rocess.job_id]=current_time;
                else{
                    current_process.current_time_io=current_time+current_proce
ss.io_time.front();
                    current_process.io_time.pop();
                    Input_Output.push_back(current_process);
                }
            }else
                ready_queue.push(current_process);
        }
        while(!Process_Arrival_Queue.empty() &&
Process_Arrival_Queue.front().second<=current_time){
            ready_queue.push(Process_Arrival_Queue.front().first);
            Process_Arrival_Queue.pop();
```

```
        }
        current_time++;
        if(ready_queue.size()==0 && Input_Output.size()==0){
            break;
        }
    }

    float avg_turnaround_time=0,avg_waiting_time=0;

    for(auto itr:Completion_Time)
    {
       cout<<"\n";
       int value1=itr.second-Arrival_Time_Map[itr.first];
       int value2=value1-mp[itr.first];
       cout<<"JobId:"<<itr.first<<"\nTurnaround Time:"<<value1<<"\nWaiting
Time:"<<value2<<"\n";
       avg_turnaround_time+=value1;
       avg_waiting_time+=value2;
       cout<<"\n";
    }
    cout<<"The average turnaround time is :"<<avg_turnaround_time/n<<"\n";
    cout<<"The average waiting time is :"<<avg_waiting_time/n<<"\n";

}
void Round_Robin(string filename)
{
   Process current_process(-1);
   queue<pair<Process,int>> Process_Arrival_Queue;
   unordered_map<int,int> Completion_Time;
   unordered_map<int ,int > Arrival_Time_Map;
   unordered_map<int,int> mp;
   priority_queue<Process> Input_Output;
   queue<Process> ready_queue;
   int n=0;

   Read_Process_Data(filename,Process_Arrival_Queue,n,Arrival_Time_Map,mp);

   for(int i=0;;){
        while(!Process_Arrival_Queue.empty() &&
Process_Arrival_Queue.front().second<=i){
```

```
            ready_queue.push(Process_Arrival_Queue.front().first);
            Process_Arrival_Queue.pop();
        }
        while(!Input_Output.empty() && Input_Output.top().current_time_io<=i){
            if(Input_Output.top().burst_time.size()==0){
                Completion_Time[Input_Output.top().job_id]=i;
            }else
                ready_queue.push(Input_Output.top());
            Input_Output.pop();
        }
        if(ready_queue.size()){
            current_process=ready_queue.front();
            ready_queue.pop();
            if(current_process.burst_time.front()<=TIMEQUANTUM){
                i+=current_process.burst_time.front();
                current_process.burst_time.pop();
                if(current_process.io_time.size()==0){
                    Completion_Time[current_process.job_id]=i;
                }else if(current_process.io_time.size()){
                    current_process.current_time_io=i+current_process.io_time.
front();
                    current_process.io_time.pop();
                    Input_Output.push(current_process);
                }
            }else{
                i+=TIMEQUANTUM;
                current_process.burst_time.front()-=TIMEQUANTUM;
                while(!Process_Arrival_Queue.empty() &&
Process_Arrival_Queue.front().second<=i){
                    ready_queue.push(Process_Arrival_Queue.front().first);
                    Process_Arrival_Queue.pop();
                }
                ready_queue.push(current_process);
            }
        }else
            i++;
        if(ready_queue.size()==0 && Input_Output.size()==0){
            break;
        }
    }
```

```
    float avg_turnaround_time=0,avg_waiting_time=0;

    for(auto itr:Completion_Time)
    {
       cout<<"\n";
       int value1=itr.second-Arrival_Time_Map[itr.first];
       int value2=value1-mp[itr.first];
       cout<<"JobId:"<<itr.first<<"\nTurnaround Time:"<<value1<<"\nWaiting
Time:"<<value2<<"\n";
       avg_turnaround_time+=value1;
       avg_waiting_time+=value2;
       cout<<"\n";
    }
    cout<<"The average turnaround time is :"<<avg_turnaround_time/n<<"\n";
    cout<<"The average waiting time is :"<<avg_waiting_time/n<<"\n";

}
int main()
{
    string filename="ProcessDataSample.txt";
    //Case 1 : First Come First Serve Algorithm
    cout<<"-------First Come First Serve -------\n";
    First_Come_First_Serve(filename);

    cout<<"\n";
    //Case 2:  Priority Scheduling Algorithm
    cout<<"---------Priority Scheduling -------\n";
    Priority_Scheduling(filename);
    cout<<"\n";
    //Case 3:  Round Robin Scheduling Algorithm

    cout<<"---------Round Robin--------\n";
    Round_Robin(filename);
    return 0;
}
```

Output:

-------First Come First Serve -------

JobId:25 Turnaround Time:2998 Waiting Time:2882

JobId:26 Turnaround Time:2988 Waiting Time:2819

JobId:18 Turnaround Time:3008 Waiting Time:2827

JobId:22 Turnaround Time:2971 Waiting Time:2832

JobId:16 Turnaround Time:2951 Waiting Time:2822

JobId:8 Turnaround Time:2998 Waiting Time:2872

JobId:9 Turnaround Time:2990 Waiting Time:2852

JobId:7 Turnaround Time:2995 Waiting Time:2809

JobId:12 Turnaround Time:2236 Waiting Time:2146

JobId:10 Turnaround Time:2190 Waiting Time:2060

JobId:11 Turnaround Time:1666 Waiting Time:1579

JobId:2 Turnaround Time:1319 Waiting Time:1252

JobId:4 Turnaround Time:363 Waiting Time:317

JobId:27 Turnaround Time:2498 Waiting Time:2406

JobId:3 Turnaround Time:1682 Waiting Time:1573

JobId:14 Turnaround Time:949 Waiting Time:901

JobId:30 Turnaround Time:1369 Waiting Time:1313

JobId:19 Turnaround Time:1053 Waiting Time:1006

JobId:5 Turnaround Time:427 Waiting Time:388

JobId:28 Turnaround Time:2458 Waiting Time:2373

JobId:15 Turnaround Time:2228 Waiting Time:2148

JobId:6 Turnaround Time:2336 Waiting Time:2233

JobId:29 Turnaround Time:2482 Waiting Time:2398

JobId:20 Turnaround Time:2312 Waiting Time:2196

JobId:1 Turnaround Time:2571 Waiting Time:2417

JobId:24 Turnaround Time:2769 Waiting Time:2636

JobId:23 Turnaround Time:2353 Waiting Time:2235

JobId:13 Turnaround Time:2711 Waiting Time:2572

JobId:17 Turnaround Time:2693 Waiting Time:2572

JobId:21 Turnaround Time:2761 Waiting Time:2630

The average turnaround time is :2244.17

The average waiting time is :2135.53

---------Priority Scheduling -------

JobId:9 Turnaround Time:3184 Waiting Time:3046

JobId:26 Turnaround Time:2978 Waiting Time:2809

JobId:3 Turnaround Time:3028 Waiting Time:2919

JobId:7 Turnaround Time:2860 Waiting Time:2674

JobId:12 Turnaround Time:2737 Waiting Time:2647

JobId:21 Turnaround Time:2567 Waiting Time:2436

JobId:11 Turnaround Time:2634 Waiting Time:2547

JobId:19 Turnaround Time:2413 Waiting Time:2366

JobId:25 Turnaround Time:875 Waiting Time:759

JobId:2 Turnaround Time:198 Waiting Time:131

JobId:28 Turnaround Time:493 Waiting Time:408

JobId:5 Turnaround Time:1478 Waiting Time:1439

JobId:22 Turnaround Time:535 Waiting Time:396

JobId:29 Turnaround Time:301 Waiting Time:217

JobId:6 Turnaround Time:2126 Waiting Time:2023

JobId:13 Turnaround Time:985 Waiting Time:846

JobId:1 Turnaround Time:325 Waiting Time:171

JobId:24 Turnaround Time:1189 Waiting Time:1056

JobId:4 Turnaround Time:314 Waiting Time:268

JobId:27 Turnaround Time:1688 Waiting Time:1596

JobId:8 Turnaround Time:1053 Waiting Time:927

JobId:30 Turnaround Time:286 Waiting Time:230

JobId:17 Turnaround Time:1332 Waiting Time:1211

JobId:23 Turnaround Time:1444 Waiting Time:1326

JobId:20 Turnaround Time:1522 Waiting Time:1406

JobId:10 Turnaround Time:1842 Waiting Time:1712

JobId:15 Turnaround Time:1993 Waiting Time:1913

JobId:14 Turnaround Time:2132 Waiting Time:2084

JobId:18 Turnaround Time:2174 Waiting Time:1993

JobId:16 Turnaround Time:2382 Waiting Time:2253

The average turnaround time is :1635.6

The average waiting time is :1526.97

---------Round Robin--------

JobId:26 Turnaround Time:2988 Waiting Time:2819

JobId:7 Turnaround Time:3154 Waiting Time:2968

JobId:18 Turnaround Time:3031 Waiting Time:2850

JobId:22 Turnaround Time:2919 Waiting Time:2780

JobId:24 Turnaround Time:2895 Waiting Time:2762

JobId:21 Turnaround Time:2901 Waiting Time:2770

JobId:25 Turnaround Time:2850 Waiting Time:2734

JobId:13 Turnaround Time:2965 Waiting Time:2826

JobId:6 Turnaround Time:2211 Waiting Time:2108

JobId:29 Turnaround Time:2551 Waiting Time:2467

JobId:11 Turnaround Time:2076 Waiting Time:1989

JobId:30 Turnaround Time:1517 Waiting Time:1461

JobId:19 Turnaround Time:1299 Waiting Time:1252

JobId:15 Turnaround Time:2038 Waiting Time:1958

JobId:4 Turnaround Time:302 Waiting Time:256

JobId:27 Turnaround Time:2526 Waiting Time:2434

JobId:3 Turnaround Time:2162 Waiting Time:2053

JobId:14 Turnaround Time:1241 Waiting Time:1193

JobId:5 Turnaround Time:429 Waiting Time:390

JobId:28 Turnaround Time:2219 Waiting Time:2134

JobId:2 Turnaround Time:1388 Waiting Time:1321

JobId:12 Turnaround Time:2479 Waiting Time:2389

JobId:17 Turnaround Time:2456 Waiting Time:2335

JobId:20 Turnaround Time:2510 Waiting Time:2394

JobId:1 Turnaround Time:2810 Waiting Time:2656

JobId:10 Turnaround Time:2754 Waiting Time:2624

JobId:16 Turnaround Time:2715 Waiting Time:2586

JobId:8 Turnaround Time:2819 Waiting Time:2693

JobId:23 Turnaround Time:2712 Waiting Time:2594

JobId:9 Turnaround Time:2943 Waiting Time:2805

The average turnaround time is :2328.67

The average waiting time is :2220.03