Institute of Engineering & Management
Department of Computer Science & Engineering
Operating System Lab for 3rd year 6th semester 2019
Code: CS 693

Date: 27/03/19

WEEK-6

Assignment-1

Problem Statement: Write a program to implement the Priority Scheduling scheduling algorithm. Use the example given in the manual as your test case.

Source Code:

```
#include <iostream>
#include <vector>
#include <tuple>
#include <algorithm>

struct Compare{
    bool operator()(std::tuple<int,int,int,int> a,
                    std::tuple<int,int,int,int> b){
        if( std::get<3>(a)<std::get<3>(b) )
            return true;
        else if( std::get<3>(a)==std::get<3>(b) )
            if( std::get<1>(a)<std::get<1>(b) )
                return true;
            else if( std::get<1>(a)==std::get<1>(b) )
                return std::get<0>(a)<std::get<0>(b);
            else return false;
        else return false;
    }
};

int main()
{
    std::cout<<"\t----Priority Scheduling----\n\n";
    int n, time=0;
    std::cout<<"Enter the No. of Processes: ";
    std::cin>>n;
    std::vector<std::tuple<int,int,int,int>> pool(n);
    std::vector<int> AT(n), CT(n), BT(n);
    std::cout<<"Arrival Time: ";
    for(int i=0;i<n;i++)
    {
        std::get<0>(pool[i]) = i+1;
        std::cin>>AT[i];
        std::get<1>(pool[i])=AT[i];
    }
    std::cout<<"Burst Time: ";
    for(int i=0;i<n;i++)
    {
        std::cin>>BT[i];
        std::get<2>(pool[i])=BT[i];
    }
    std::cout<<"Priority(1 is highest): ";
    for(int i=0;i<n;i++)
        std::cin>>std::get<3>(pool[i]);

    std::vector<std::tuple<int,int,int,int>> current;
    while(true)
    {
```

```
        for(auto it=pool.begin();it!=pool.end();it++)
            if(time >= std::get<1>(*it) && std::get<0>(*it)!=0)
            {
                current.push_back(*it);
                std::get<0>(*it) = 0;
            }

        std::stable_sort(current.begin(),current.end(), Compare());
        if(!current.empty())
        {
            std::get<2>(current[0])--;
            if(std::get<2>(current[0]) == 0)
            {
                CT[std::get<0>(current[0])-1] = time+1;
                current.erase(current.begin());
            }
        }
        time++;

        bool v = false;
        for(int i=0;i<n;i++)
            if(std::get<0>(pool[i]) != 0)
                v = v || true;
        v = v || !current.empty();
        if(!v)
            break;
    }

    std::cout<<"\nPID\tArrival Time\tCompletion Time\tBurst Time\tWaiting
                                                        Time\n";
    int avgWT=0, avgTAT=0;
    for(int i=0;i<n;i++)
    {
        avgWT+=CT[i]-AT[i]-BT[i];
        avgTAT+=CT[i]-AT[i];
        std::cout<<i+1<<"\t"<<AT[i]<<"\t\t"<<CT[i]<<"\t\t"<<BT[i]<<"\t\t"<<
                                    CT[i]-AT[i]-BT[i]<<"\n";
    }
    std::cout<<"\nAverage Waiting Time: "<<avgWT/n;
    std::cout<<"\n\nAverage Turn Around Time: "<<avgTAT/n<<"\n\n";
}
```

Screen-Shot:

```
iemcse@ubuntu:~/Desktop/rana$ ./a.out
        ----Priority Scheduling----

Enter the No. of Processes: 5
Arrival Time: 0 0 0 0 0
Burst Time: 10 1 2 1 5
Priority(1 is highest): 3 1 4 5 2

PID     Arrival Time    Completion Time Burst Time      Waiting Time
1       0               16              10              6
2       0               1               1               0
3       0               18              2               16
4       0               19              1               18
5       0               6               5               1

Average Waiting Time: 8

Average Turn Around Time: 12
```

Assignment-2

Problem Statement: Write a program to implement the Round robin scheduling algorithm. Use the example given in the manual as your test case.

Source Code:

```
#include <iostream>
#include <vector>
#include <tuple>
#include <algorithm>
#include <queue>

struct Compare{
    bool operator()(std::tuple<int,int,int> a, std::tuple<int,int,int> b){
        if( std::get<1>(a)<std::get<1>(b) )
            return true;
        else if( std::get<1>(a)==std::get<1>(b) )
            return std::get<0>(a)<std::get<0>(b);
        else return false;
    }
};

int main()
{
    std::cout<<"\t----Round Robin Scheduling----\n\n";
    int n, time=0, rt=0, tq;
    std::cout<<"Enter the No. of Processes: ";
    std::cin>>n;
    std::vector<std::tuple<int,int,int>> pool(n);
    std::queue<std::tuple<int,int,int>> cur_q;
    std::vector<int> AT(n), CT(n), BT(n);
    std::cout<<"Arrival Time: ";
    for(int i=0;i<n;i++)
    {
        std::get<0>(pool[i]) = i+1;
        std::cin>>AT[i];
        std::get<1>(pool[i])=AT[i];
    }
    std::cout<<"Burst Time: ";
    for(int i=0;i<n;i++)
    {
        std::cin>>BT[i];
        std::get<2>(pool[i])=BT[i];
    }
    std::cout<<"Time Quantum: ";
    std::cin>>tq;
    std::stable_sort(pool.begin(), pool.end(), Compare());
    std::tuple<int,int,int> current;
    while(true)
    {
        for(auto it=pool.begin();it!=pool.end();it++)
            if(time >= std::get<1>(*it) && std::get<0>(*it)!=0)
            {
                cur_q.push(*it);
                std::get<0>(*it) = 0;
            }
        if(!cur_q.empty())
        {
            std::get<2>(cur_q.front())--;
            rt++;
            if(std::get<2>(cur_q.front()) == 0)
            {
                CT[std::get<0>(cur_q.front())-1] = time+1;
                cur_q.pop();
```

```
                rt=0;
            }
        }
        time++;
        if(rt==tq)
        {
            cur_q.push(cur_q.front());
            cur_q.pop();
            rt=0;
        }
        bool v = false;
        for(int i=0;i<n;i++)
            if(std::get<0>(pool[i]) != 0)
                v = v || true;
        v = v || !cur_q.empty();
        if(!v)
            break;
    }

    std::cout<<"\nPID\tArrival Time\tCompletion Time\tBurst Time\tWaiting
                                                            Time\n";
    int avgWT=0, avgTAT=0;
    for(int i=0;i<n;i++)
    {
        avgWT+=CT[i]-AT[i]-BT[i];
        avgTAT+=CT[i]-AT[i];
        std::cout<<i+1<<"\t"<<AT[i]<<"\t\t"<<CT[i]<<"\t\t"<<BT[i]<<"\t\t"
                                        <<CT[i]-AT[i]-BT[i]<<"\n";
    }
    std::cout<<"\nAverage Waiting Time: "<<avgWT/n;
    std::cout<<"\n\nAverage Turn Around Time: "<<avgTAT/n<<"\n\n";
}
```

Screen-Shot:

```
iemcse@ubuntu:~/Desktop/rana$ g++ rr.cpp
iemcse@ubuntu:~/Desktop/rana$ ./a.out
        ----Round Robin Scheduling----

Enter the No. of Processes: 3
Arrival Time: 0 0 0
Burst Time: 24 3 3
Time Quantum: 4

PID     Arrival Time    Completion Time Burst Time      Waiting Time
1       0               30              24              6
2       0               7               3               4
3       0               10              3               7

Average Waiting Time: 5

Average Turn Around Time: 15
```