Name: Brijesh Rameshbhai Rohit

Admission number: U19CS009

OS-ASSIGNMENT-06

Write a program for the simulation of

1. Shortest Job First (SJF)

2. Shortest Remaining Time First (SRTF) CPU scheduler.

3. Round Robin Scheduling.

Take n no. of process from user with arrival_time and burst_time. Arrival_time and burst_time should be generated randomly and compute Completion Time, Turnaround (TAT) Time and Waiting Time. Also show the count of context switching in all of the algorithms.

1. Shortest Job First (SJF)

Code :

```
#include <bits/stdc++.h>
    using namespace std;
class process
{
public:
    int processId;
    int arrivalTime;
    int burstTime;
    int completionTime;
    int turnarondTime;
    int waitingTime;
    process(int processId, int a, int b)
    {
        this->processId = processId;
        this->arrivalTime = a;
        this->burstTime = b;
    }
};
//function to compare process
bool cmp(const process &obj1, const process &obj2)
{
    if (obj1.arrivalTime == obj2.arrivalTime)
    {
        return obj1.burstTime < obj2.burstTime;
    }
    return obj1.arrivalTime < obj2.arrivalTime;
```

```
}
//Shortest job first
void sjf(vector<process> &v, int n)
{
    int tcomp, curr;
    // First process will complete
    v[0].completionTime = v[0].arrivalTime + v[0].burstTime;
    v[0].turnarondTime = v[0].completionTime - v[0].arrivalTime;
    v[0].waitingTime = v[0].turnarondTime - v[0].burstTime;
    for (int i = 1; i < n; i++)
    {
        tcomp = v[i - 1].completionTime;
        int min_burst = v[i].burstTime;
        curr = -1;
        while (curr == -1)
        {
            for (int j = i; j < n; j++)
            {
                if (tcomp >= v[j].arrivalTime && min_burst >= v[j].burstTime)
                {
                    min_burst = v[j].burstTime;
                    curr = j;
                }
            }
            if (curr == -1)
                tcomp = v[i].arrivalTime;
        }
        v[curr].completionTime = tcomp + v[curr].burstTime;
        v[curr].turnarondTime = v[curr].completionTime - v[curr].arrivalTime;
        v[curr].waitingTime = v[curr].turnarondTime - v[curr].burstTime;
        process temp = v[i];
        v[i] = v[curr];
        v[curr] = temp;
    }
}
// function to generate random input
vector<process> randominput(int n)
{
    unsigned seed = chrono::system_clock::now().time_since_epoch().count();
    default_random_engine generator(seed);
    uniform_int_distribution<int> d1(0, ((10 * n) + 1) / 2);
    uniform_int_distribution<int> d2(1, 10);
    vector<process> input;
    for (int i = 0; i < n; i++)
    {
```

```
        input.push_back(process(i + 1, d1(generator), d2(generator)));
        d1.reset();
    }
    return input;
}
int main()
{
    int n;
    cout << "Enter Number of Processes : ";
    cin >> n;
    vector<process> v1;
    //random generation
    v1 = randominput(n);
    cout << "\nRandomly generated inputs are : \n";
    cout << "\nProcess Id\tArrival Time\tBurst Time\n";
    for (int i = 0; i < n; i++)
    {
        cout << v1[i].processId << "\t\t"
             << v1[i].arrivalTime << "\t\t" << v1[i].burstTime << "\n";
    }
    //sorting process according to arrival time
    sort(v1.begin(), v1.end(), cmp);
    sjf(v1, n);
    int tot_wt = 0, tot_tt = 0;
    cout << "\n\nProcess Id\tArrival Time\tBurst Time\t Completion
Time\tWaitingTime\tTurn Around Time\n";
        for (int i = 0; i < n; i++)
    {
        cout << v1[i].processId << "\t\t" << v1[i].arrivalTime << "\t\t" <<
v1[i].burstTime << "\t\t " << v1[i].completionTime << "\t\t\t" <<
v1[i].waitingTime << "\t\t " << v1[i].turnarondTime << "\n";
        tot_wt += v1[i].waitingTime;
        tot_tt += v1[i].turnarondTime;
    }
    cout << "\n\nAverage Waiting time : " << (float)tot_wt / (float)n;
    cout << "\nAverage Turn Around time : " << (float)tot_tt / (float)n;
}
```

Ouput :

```
Enter Number of Processes : 5

Randomly generated inputs are :

Process Id      Arrival Time    Burst Time
1               12              8
2               21              5
3               24              4
4               3               6
5               8               5


Process Id      Arrival Time    Burst Time      Completion Time         WaitingTime     Turn Around Time
4               3               6               9                       0               6
5               8               5               14                      1               6
1               12              8               22                      2               10
2               21              5               27                      1               6
3               24              4               31                      3               7


Average Waiting time : 1.4
Average Turn Around time : 7
```

2. Shortest Remaining Time First (SRTF) CPU scheduler.

Code :

```cpp
#include <bits/stdc++.h>
using namespace std;
class process
{
public:
    int processId;
    int arrivalTime;
    int burstTime;
    int completionTime;
    int turnarondTime;
    int waitingTime;
    int remainingTime;
    process(int processId, int a, int b)
    {
        this->processId = processId;
        this->arrivalTime = a;
        this->burstTime = b;
    }
};
//function to compare process
bool cmp(const process &obj1, const process &obj2)
{
    if (obj1.arrivalTime == obj2.arrivalTime)
    {
        return obj1.burstTime < obj2.burstTime;
    }
```

```
    return obj1.arrivalTime < obj2.arrivalTime;
}
// Shortest Remaining Time First
void srtf(vector<process> &v, int n)
{
    //remaing time = burst time initially
    for (int i = 0; i < n; i++)
    {
        v[i].remainingTime = v[i].burstTime;
    }
    int context_switch = 0;
    int compnum = 0, ctime = 0, minrem_time = INT_MAX;
    int minproc = 0, tcomp;
    bool flag = false;
    string s = "", s1 = "";
    while (compnum != n)
    {
        s1 += to_string(ctime) + " ";
        if (ctime < 10)
            s1 += " ";
        bool flag1 = false;
        for (int i = 0; i < n; i++)
        {
            //condition for minimum remaining time
            if ((v[i].arrivalTime <= ctime) && (v[i].remainingTime <
minrem_time) && (v[i].remainingTime > 0))
            {
                //a new process is executed no context switching
                if (minrem_time == INT_MAX)
                {
                    flag1 = true;
                }
                minrem_time = v[i].remainingTime;
                minproc = i;
                flag = true;
                if (flag1 == true)
                    continue;
                context_switch++;
            }
        }
        //No process in queue with minimum remaining time
        if (flag == false)
        {
            if (minrem_time == INT_MAX)
                s += " ";
```

```
            else
                s += "P" + to_string(v[minproc].processId) + " ";
            ctime++;
            continue;
        }
        s += "P" + to_string(v[minproc].processId) + " ";
        //decrementing remaining time of the feasable process
        v[minproc].remainingTime--;
        minrem_time = v[minproc].remainingTime; //updating minrem_time
        if (minrem_time == 0)                   //process is complete thus
reseting
        {
            minrem_time = INT_MAX;
        }
        if (v[minproc].remainingTime == 0) //process is complete so caculating
other times
        {
            compnum++; //completed process incremented
            flag = false;
            tcomp = ctime + 1; //completion time = current time + 1;
            //updating data members
            v[minproc].completionTime = tcomp;
            v[minproc].waitingTime = tcomp - v[minproc].burstTime -
v[minproc].arrivalTime;
            if (v[minproc].waitingTime < 0)
            {
                v[minproc].waitingTime = 0;
            }
            v[minproc].turnarondTime = v[minproc].waitingTime +
v[minproc].burstTime;
        }
        ctime++;
    }
    cout
        << "\n\nTotal number of context switching : " << context_switch;
}
//function to generated random inputs
vector<process> randominput(int n)
{
    unsigned seed = chrono::system_clock::now().time_since_epoch().count();
    default_random_engine generator(seed);
    uniform_int_distribution<int> d1(0, ((10 * n) + 1) / 2);
    uniform_int_distribution<int> d2(1, 10);
    vector<process> input;
    for (int i = 0; i < n; i++)
```

```
    {
        input.push_back(process(i + 1, d1(generator), d2(generator)));
        d1.reset();
    }
    return input;
}
int main()
{
    int n;
    cout << "Enter Number of Processes : ";
    cin >> n;
    vector<process> v;
    //random inputs
    v = randominput(n);
    cout << "Random process genrated are : ";
    cout << "\nProcess Id\tArrival Time\tBurst Time\n";
    for (int i = 0; i < n; i++)
    {
        cout << v[i].processId << "\t\t"
             << v[i].arrivalTime << "\t\t" << v[i].burstTime << "\n";
    }
    //sorting according to arrival time
    sort(v.begin(), v.end(), cmp);
    srtf(v, n);
    cout << "\n\nProcess Id\tArrival Time\tBurst Time\t Completion
Time\tWaiting Time\tTurn Around Time\n";
        int tot_wt = 0,
            tot_tt = 0;
    for (int i = 0; i < n; i++)
    {
        cout << v[i].processId << "\t\t" << v[i].arrivalTime << "\t\t" <<
v[i].burstTime << "\t\t" << v[i].completionTime << "\t\t\t" << v[i].waitingTime
<< "\t\t" << v[i].turnarondTime << "\n";
        tot_wt += v[i].waitingTime;
        tot_tt += v[i].turnarondTime;
    }
    cout << "\n\nAverage Waiting time : " << (float)tot_wt / (float)n;
    cout << "\nAverage Turn Around time : " << (float)tot_tt / (float)n;
}
```

Output :

```
Enter Number of Processes : 6
Random process genrated are :
Process Id      Arrival Time    Burst Time
1               14              4
2               14              6
3               18              10
4               16              10
5               27              8
6               23              2


Total number of context switching : 1

Process Id      Arrival Time    Burst Time       Completion Time        Waiting Time    Turn Around Time
1               14              4               18                      0               4
2               14              6               24                      4               10
4               16              10              44                      18              28
3               18              10              54                      26              36
6               23              2               26                      1               3
5               27              8               35                      0               8


Average Waiting time : 8.16667
Average Turn Around time : 14.8333
```

3. Round Robin Scheduling.

Code :

```cpp
#include <bits/stdc++.h>
using namespace std;
class process
{
public:
    int processId;
    int arrivalTime;
    int burstTime;
    int completionTime;
    int turnarondTime;
    int waitingTime;
    int remainingTime;
    process(int processId, int a, int b)
    {
        this->processId = processId;
        this->arrivalTime = a;
        this->burstTime = b;
    }
};
//function to compare process
bool cmp(const process &obj1, const process &obj2)
{
    if (obj1.arrivalTime == obj2.arrivalTime)
    {
```

```
        return obj1.burstTime < obj2.burstTime;
    }
    return obj1.arrivalTime < obj2.arrivalTime;
}
void rrs(vector<process> &v, int n, int interval)
{
    int ctime = 0;
    //remaining time = burst time
    for (int i = 0; i < n; i++)
    {
        v[i].remainingTime = v[i].burstTime;
    }
    int comproc = 0;
    string s1 = "Time : ";
    string s2 = "Process : ";
    int context_switch = 0;
    int prev_i = -1;
    while (comproc != n)
    {
        bool flag = true;
        for (int i = 0; i < n; i++)
        {
            //if any process has arrived
            if (v[i].arrivalTime <= ctime)
            {
                // process is incomplete
                if (v[i].remainingTime > 0)
                {
                    flag = false;
                    if (prev_i != i && prev_i != -1)
                    {
                        if (v[prev_i].remainingTime != 0)
                        {
                            context_switch++;
                        }
                    }
                    prev_i = i;
                    if (v[i].remainingTime > interval)
                    {
                        ctime += interval;
                        v[i].remainingTime -= interval;
                        for (int j = 0; j < interval; j++)
                        {
                            s1 += "P" + to_string(v[i].processId) + " ";
                        }
```

```
                    }
                    else
                    {
                        for (int j = 0; j < v[i].remainingTime; j++)
                        {
                            s1 += "P" + to_string(v[i].processId) + " ";
                        }
                        ctime += v[i].remainingTime;
                        v[i].waitingTime = ctime - v[i].burstTime -
v[i].arrivalTime;
                        v[i].remainingTime = 0;
                        v[i].completionTime = ctime;
                        //increment the completed process number
                        comproc++;
                        prev_i = i;
                    }
                }
            }
        }
        if (flag == true)
        {
            ctime++;
            s1 += " ";
        }
    }
    for (int i = 0; i <= ctime; i++)
    {
        s2 += to_string(i) + " ";
        if (i < 10)
            s2 += " ";
    }
    for (int i = 0; i < n; i++)
    {
        v[i].turnarondTime = v[i].burstTime + v[i].waitingTime;
    }
    cout << "\n\nTotal number of context switching : " << context_switch;
}
//function for random input generation
vector<process> randominput(int n)
{
    unsigned seed = chrono::system_clock::now().time_since_epoch().count();
    default_random_engine generator(seed);
    uniform_int_distribution<int> d1(1, ((10 * n) + 1) / 2);
    uniform_int_distribution<int> d2(1, 10);
    vector<process> input;
```

```
    for (int i = 0; i < n; i++)
    {
        input.push_back(process(i + 1, d1(generator), d2(generator)));
        d1.reset();
    }
    return input;
}
int main()
{
    int n;
    cout << "Enter Number of Processes : ";
    cin >> n;
    int interval;
    cout << "Enter the time quantum for round robin : ";
    cin >> interval;
    vector<process> v1;
    //random input
    v1 = randominput(n);
    cout << "\nRandomly generated inputs are : ";
    cout << "\nProcess Id\tArrival Time\tBurst Time\n";
    for (int i = 0; i < n; i++)
    {
        cout << v1[i].processId << "\t\t"
             << v1[i].arrivalTime << "\t\t" << v1[i].burstTime << "\n";
    }
    //sorting according to arrive time
    sort(v1.begin(), v1.end(), cmp);
    rrs(v1, n, interval);
    int tot_wt = 0, tot_tt = 0;
    cout << "\n\nProcess ID\tArrival Time\tBurst Time\t Completion
Time\tWaiting Time\tTurn Around Time\n";
        for (int i = 0; i < n; i++)
    {
        cout << "\t" << v1[i].processId << "\t\t" << v1[i].arrivalTime <<
"\t\t" << v1[i].burstTime << "\t\t "
             << v1[i].completionTime << "\t\t\t " << v1[i].waitingTime << "\t\t
" << v1[i].turnarondTime << endl;
        tot_wt += v1[i].waitingTime;
        tot_tt += v1[i].turnarondTime;
    }
    cout << "\n\nAverage Waiting time : " << (float)tot_wt / (float)n;
    cout << "\nAverage Turn Around time : " << (float)tot_tt / (float)n;
}
```

Output :

```
Enter Number of Processes : 6
Enter the time quantum for round robin : 3

Randomly generated inputs are :
Process Id      Arrival Time    Burst Time
1               25              9
2               3               9
3               21              8
4               10              7
5               18              7
6               8               7


Total number of context switching : 11

Process ID      Arrival Time    Burst Time      Completion Time         Waiting Time    Turn Around Time
        2               3               9               18                      6               15
        6               8               7               34                      19              26
        4               10              7               35                      18              25
        5               18              7               45                      20              27
        3               21              8               47                      18              26
        1               25              9               50                      16              25


Average Waiting time : 16.1667
Average Turn Around time : 24
```