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-5

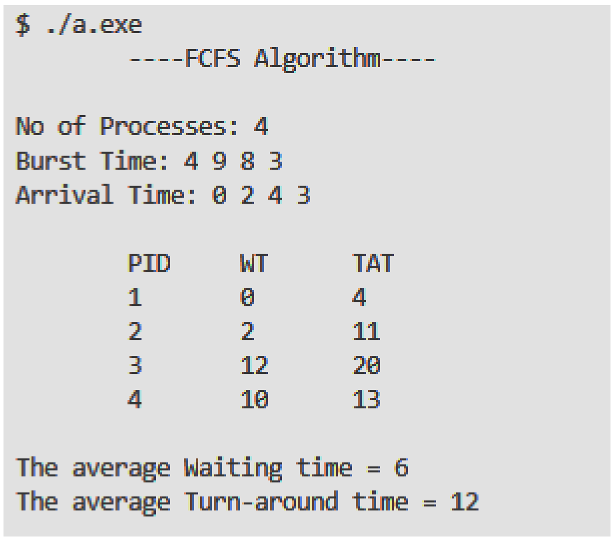
Assignment-1

Problem Statement: Write a program to implement the FCFS scheduling algorithm.

Source Code:
```
#include <iostream>
#include <queue>
#include <tuple>
#include <algorithm>

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()
{
    int n;
    std::cout<<"\t----FCFS Algorithm----\n\nNo of Processes: ";
    std::cin>>n;
    std::vector<std::tuple<int,int,int>> atbt(n);
    std::vector<int> rt(n+1), ct(n+1), tat(n+1), wt(n+1);
    std::priority_queue<std::tuple<int,int,int>,
std::vector<std::tuple<int,int,int>>, Compare> q, ready;

    std::cout<<"Burst Time: ";
    for(auto i=0; i<n; i++)
    {
        int temp;
        std::get<0>(atbt[i]) = i+1;
        std::cin>>temp;
        std::get<2>(atbt[i]) = temp;
    }

    std::cout<<"Arrival Time: ";
    for(auto i=0; i<n; i++)
    {
        int temp;
        std::cin>>temp;
        std::get<1>(atbt[i]) = temp;
        q.push(atbt[i]);
    }

    int counter=0, cpid=0;
    std::tuple<int,int,int> current_p(0,0,0);
    rt[0] = 0;
    do{
        while(counter == std::get<1>(q.top()) && !q.empty())
```

```
            {
                ready.push(q.top());
                q.pop();
            }
            if(std::get<2>(current_p) == counter - rt[cpid] && cpid!=0)
            {
                ct[cpid] = counter;
                tat[cpid] = counter - std::get<1>(current_p);
                wt[cpid] = tat[cpid] - std::get<2>(current_p);
                current_p = std::make_tuple(0,0,0);
                cpid = 0;
            }
            if(cpid==0)
            {
                if(!ready.empty())
                {
                    current_p = ready.top();
                    ready.pop();
                    cpid = std::get<0>(current_p);
                    rt[cpid] = counter;
                }
            }
            counter++;
        }while(!q.empty() || !ready.empty() || cpid!=0);

        int avg_wt=0, avg_tat=0;
        std::cout<<"\n\tPID\tWT\tTAT\n";
        for(int i=1; i<=n; i++)
        {
            std::cout<<"\t"<<i<<"\t"<<wt[i]<<"\t"<<tat[i]<<"\n";
            avg_wt+=wt[i];
            avg_tat+=tat[i];
        }
        std::cout<<"\nThe average Waiting time = "<<(float)avg_wt/n<<"\n";
        std::cout<<"The average Turn-around time =
                                               "<<(float)avg_tat/n<<"\n\n";
    }
```

Screen-Shot:



```
$ ./a.exe
        ----FCFS Algorithm----

No of Processes: 4
Burst Time: 4 9 8 3
Arrival Time: 0 2 4 3

        PID     WT      TAT
        1       0       4
        2       2       11
        3       12      20
        4       10      13

The average Waiting time = 6
The average Turn-around time = 12
```

Assignment-2

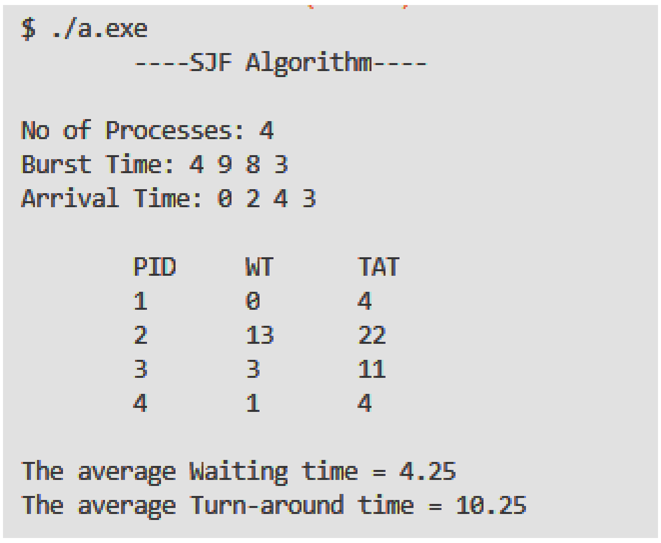
Problem Statement: Write a program to implement the SJF scheduling algorithm.

Source Code:

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

struct Compare_BT{
    bool operator()(std::tuple<int,int,int> a, std::tuple<int,int,int> b)
    {
        if(std::get<2>(a) > std::get<2>(b))
            return true;
        else if(std::get<2>(a) == std::get<2>(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;
    }
};

struct Compare_AT{
    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()
{
    int n;
    std::cout<<"\t----SJF Algorithm----\n\nNo of Processes: ";
    std::cin>>n;
    std::vector<std::tuple<int,int,int>> atbt(n);
    std::vector<int> rt(n+1), ct(n+1), tat(n+1), wt(n+1);
    std::priority_queue<std::tuple<int,int,int>,
              std::vector<std::tuple<int,int,int>>, Compare_BT> ready;
    std::priority_queue<std::tuple<int,int,int>,
              std::vector<std::tuple<int,int,int>>, Compare_AT> q;
    std::cout<<"Burst Time: ";
    for(auto i=0; i<n; i++)
    {
        int temp;
        std::get<0>(atbt[i]) = i+1;
        std::cin>>temp;
        std::get<2>(atbt[i]) = temp;
    }
    std::cout<<"Arrival Time: ";
    for(auto i=0; i<n; i++)
    {
        int temp;
        std::cin>>temp;
        std::get<1>(atbt[i]) = temp;
```

```
            q.push(atbt[i]);
        }
        int counter=0, cpid=0;
        std::tuple<int,int,int> current_p(0,0,0);
        rt[0] = 0;
        do{
            while(counter == std::get<1>(q.top()) && !q.empty())
            {
                ready.push(q.top());
                q.pop();
            }
            if(std::get<2>(current_p) == counter - rt[cpid] && cpid!=0)
            {
                ct[cpid] = counter;
                tat[cpid] = counter - std::get<1>(current_p);
                wt[cpid] = tat[cpid] - std::get<2>(current_p);
                current_p = std::make_tuple(0,0,0);
                cpid = 0;
            }
            if(cpid==0)
            {
                if(!ready.empty())
                {
                    current_p = ready.top();
                    ready.pop();
                    cpid = std::get<0>(current_p);
                    rt[cpid] = counter;
                }
            }
            counter++;
        }while(!q.empty() || !ready.empty() || cpid!=0);

        int avg_wt=0, avg_tat=0;
        std::cout<<"\n\tPID\tWT\tTAT\n";
        for(int i=1; i<=n; i++)
        {
            std::cout<<"\t"<<i<<"\t"<<wt[i]<<"\t"<<tat[i]<<"\n";
            avg_wt+=wt[i];
            avg_tat+=tat[i];
        }
        std::cout<<"\nThe average Waiting time = "<<(float)avg_wt/n<<"\n";
        std::cout<<"The average Turn-around time =
                                        "<<(float)avg_tat/n<<"\n\n";
    }
```

Screen-Shot:



```
$ ./a.exe
        ----SJF Algorithm----

No of Processes: 4
Burst Time: 4 9 8 3
Arrival Time: 0 2 4 3

        PID     WT      TAT
        1       0       4
        2       13      22
        3       3       11
        4       1       4

The average Waiting time = 4.25
The average Turn-around time = 10.25
```