OPERATING SYSTEMS
MC301 Lab

ANEESH PANCHAL
2K20/MC/21

Department of Applied Mathematics,

Delhi Technological University

Submitted To –

Dr. Anshul Arora

INDEX

S. No.	Experiment	Date	Sign & Remark
01.	Implement First Come First Serve (FCFS) job scheduling algorithm.	10/Aug/2022	
02.	Implement Shortest Job First (SJF) scheduling algorithm (non-preemptive version).	17/Aug/2022	
03.	Implement Shortest Job First (SJF) scheduling algorithm (preemptive version).	24/Aug/2022	
04.	Implement Round Robin Scheduling Algorithm (with any time quantum of your choice).	07/Sept/2022	
05.	Implement Priority based Job scheduling algorithm (non-preemptive version).	12/Oct/2022	
06.	Write a program to implement Banker's Algorithm for Deadlock Avoidance.	19/Oct/2022	
07.	Write a program to implement FCFS Disk Scheduling Algorithm.	02/Nov/2022	
08.	Write a program to implement SSTF Disk Scheduling Algorithm.	09/Nov/2022	

Aneesh Panchal
2K20/MC/21

Experiment I

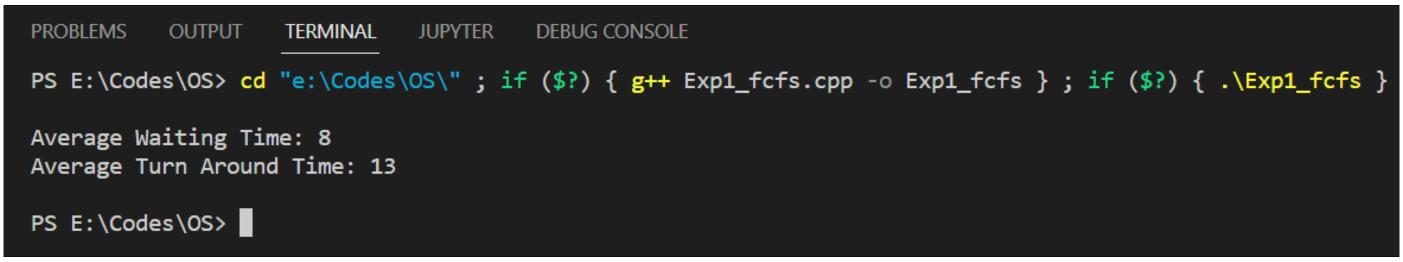
Aim:

Implement First Come First Serve (FCFS) job scheduling algorithm.

Code:

```
//Aneesh Panchal
//2K20/MC/21

#include<iostream>
#include<vector>
using namespace std;

void tut(vector<int> WT, vector<int> ET){
    vector<int> TUT(WT.size(),0);
    float avgtut = 0;
    for(int i=0;i<WT.size();i++){
        TUT[i]=WT[i]+ET[i];
        avgtut = avgtut + TUT[i];
    }
    avgtut = float(avgtut/float(WT.size()));
    cout<<"Average Turn Around Time: "<<avgtut<<endl;
}

void wt(vector<int> AT, vector<int> ET){
    vector<int> WT(AT.size(),0);
    int newET = AT[0];
    for(int i=1;i<AT.size();i++){
        newET = ET[i-1] + newET;
        WT[i] = newET - AT[i];
    }
    float avgwt = 0;
    for(int i=0;i<AT.size();i++)
        avgwt = avgwt + WT[i];
    avgwt = float(avgwt/float(WT.size()));
    cout<<"Average Waiting Time: "<<avgwt<<endl;
    tut(WT,ET);
}

int main(){
    vector<int> AT = {1,2,4,6};
    vector<int> ET = {10,3,5,2};
    cout<<endl;
    wt(AT,ET);
    cout<<endl;
    return 0;
}
```

Output:



```
PS E:\Codes\OS> cd "e:\Codes\OS\" ; if ($?) { g++ Exp1_fcfs.cpp -o Exp1_fcfs } ; if ($?) { .\Exp1_fcfs }

Average Waiting Time: 8
Average Turn Around Time: 13

PS E:\Codes\OS> 
```

Aneesh Panchal
2K20/MC/21

Experiment II

Aim:

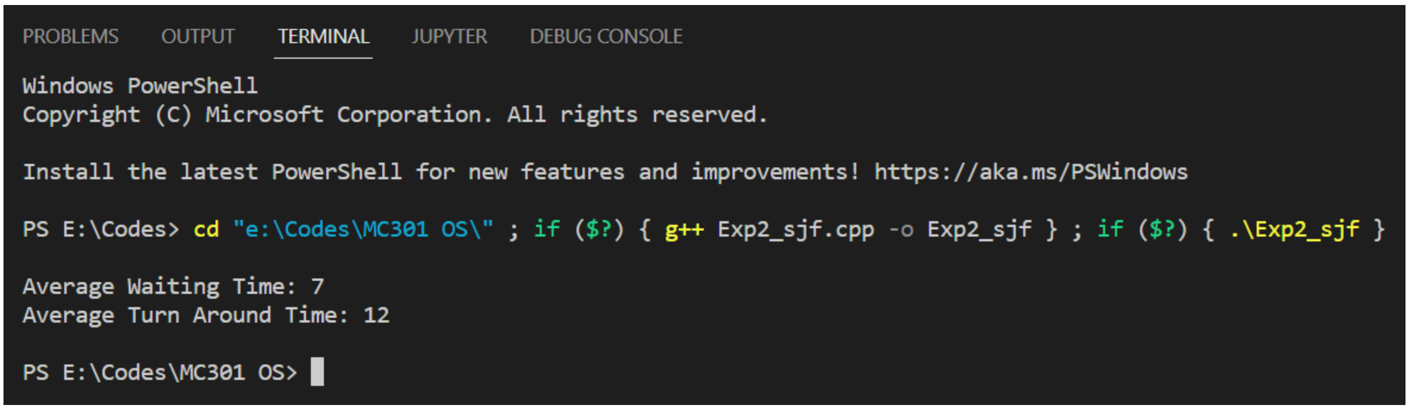
Implement Shortest Job First (SJF) scheduling algorithm (non-preemptive version).

Code:

```cpp
//Aneesh Panchal
//2K20/MC/21

#include<iostream>
#include<vector>
using namespace std;

void tut(vector<int> WT, vector<int> ET){
    vector<int> TUT(WT.size(),0);
    float avgtut = 0;
    for(int i=0;i<WT.size();i++){
        TUT[i]=WT[i]+ET[i];
        avgtut = avgtut + TUT[i];
    }
    avgtut = float(avgtut/float(WT.size()));
    cout<<"Average Turn Around Time: "<<avgtut<<endl;
}

int minarr(vector<int> AT ,vector<int> ET, vector<bool> done, int n){
    int min = 0;
    int doi = 0;
    if(n==0)
        for(int i=0;AT[0]!=n;++i)
            ++n;
    for(int i=0;i<ET.size();i++)
        if(done[i]==false){
            if(n>=AT[i])
                if(ET[i]<ET[min])
                    min = i;
            else if(doi==0 && n<AT[i]){
                n = AT[i];
                min = i;
                for(int j=i;j<ET.size()-i;++j){
                    if(done[j]==false && ET[j]<ET[min])
                        min = j;
                }
                break;
            }
            ++doi;
        }
    return min;
}
```

```cpp
void wt(vector<int> AT, vector<int> ET){
    vector<int> WT(AT.size(),0);
    vector<bool> done(AT.size(),false);
    int newET = 0;
    int index = 0;
    for(int i=0;i<AT.size();i++){
        index = minarr(AT,ET,done,newET);
        if(i==0){
            newET = AT[index] + ET[index];
            WT[index] = 0;
        }
        else{
            WT[index] = newET - AT[index];
            newET = ET[index] + newET;
            if(WT[index]<=0){
                newET = newET - WT[index];
                WT[index]=0;
            }
        }
        done[index] = true;
    }
    float avgwt = 0;
    for(int i=0;i<AT.size();i++)
        avgwt = avgwt + WT[i];
    avgwt = float(avgwt/float(WT.size()));
    cout<<"Average Waiting Time: "<<avgwt<<endl;
    tut(WT,ET);
}

int main(){
    vector<int> AT = {1,2,4,6};
    vector<int> ET = {10,3,5,2};
    cout<<endl;
    wt(AT,ET);
    cout<<endl;
    return 0;
}
```

Output:



```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS E:\Codes> cd "e:\Codes\MC301 OS\" ; if ($?) { g++ Exp2_sjf.cpp -o Exp2_sjf } ; if ($?) { .\Exp2_sjf }

Average Waiting Time: 7
Average Turn Around Time: 12

PS E:\Codes\MC301 OS>
```

Aneesh Panchal
2K20/MC/21

Experiment III

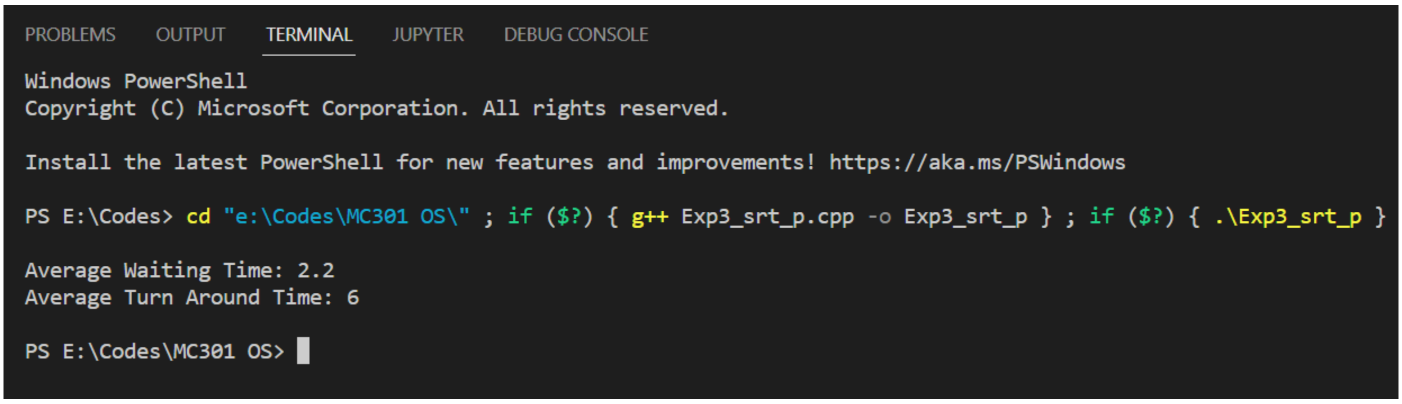
Aim:

Implement Shortest Job First (SJF) scheduling algorithm (preemptive version).

Code:

```cpp
//Aneesh Panchal
//2K20/MC/21

#include<iostream>
#include<vector>
#include<climits>
using namespace std;

void tut(vector<int> BT,vector<int> WT){
    vector<int> TUT(WT.size(),0);
    for(int i=0;i<WT.size();i++)
        TUT[i] = BT[i] + WT[i];
    float avgtut = 0;
    for(int i=0;i<WT.size();i++)
        avgtut = avgtut + TUT[i];
    avgtut = float(avgtut/float(WT.size()));
    cout<<"Average Turn Around Time: "<<avgtut<<endl;
}

void wt(vector<int> AT,vector<int> BT){
    vector<int> WT(AT.size(),0);
    vector<int> RT=BT;
    int complete = 0, t = 0, minm = INT_MAX;
    int index = 0, finish_time;
    bool check = false;
    while(complete != AT.size()){
        for(int j=0;j<AT.size();j++){
            if((AT[j]<=t) && (RT[j]<minm) && RT[j]>0){
                minm = RT[j];
                index = j;
                check = true;
            }
        }
        if(check == false){
            t++;
            continue;
        }
        RT[index]--;
        minm = RT[index];
        if(minm == 0)
            minm = INT_MAX;
        if(RT[index] == 0){
            complete++;
            check = false;
            finish_time = t + 1;
            WT[index] = finish_time - BT[index] - AT[index];
            if(WT[index]<0)
                WT[index] = 0;
        }
        t++;
    }
    float avgwt = 0;
    for(int i=0;i<AT.size();i++)
        avgwt = avgwt + WT[i];
    avgwt = float(avgwt/float(WT.size()));
    cout<<"Average Waiting Time: "<<avgwt<<endl;
    tut(BT,WT);
}

int main(){
    vector<int> AT = {0,1,2,3,4};
    vector<int> BT = {10,1,2,1,5};
    cout<<endl;
    wt(AT,BT);
    cout<<endl;
    return 0;
}
```

Output:



```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS E:\Codes> cd "e:\Codes\MC301 OS\" ; if ($?) { g++ Exp3_srt_p.cpp -o Exp3_srt_p } ; if ($?) { .\Exp3_srt_p }

Average Waiting Time: 2.2
Average Turn Around Time: 6

PS E:\Codes\MC301 OS> 
```

Aneesh Panchal
2K20/MC/21

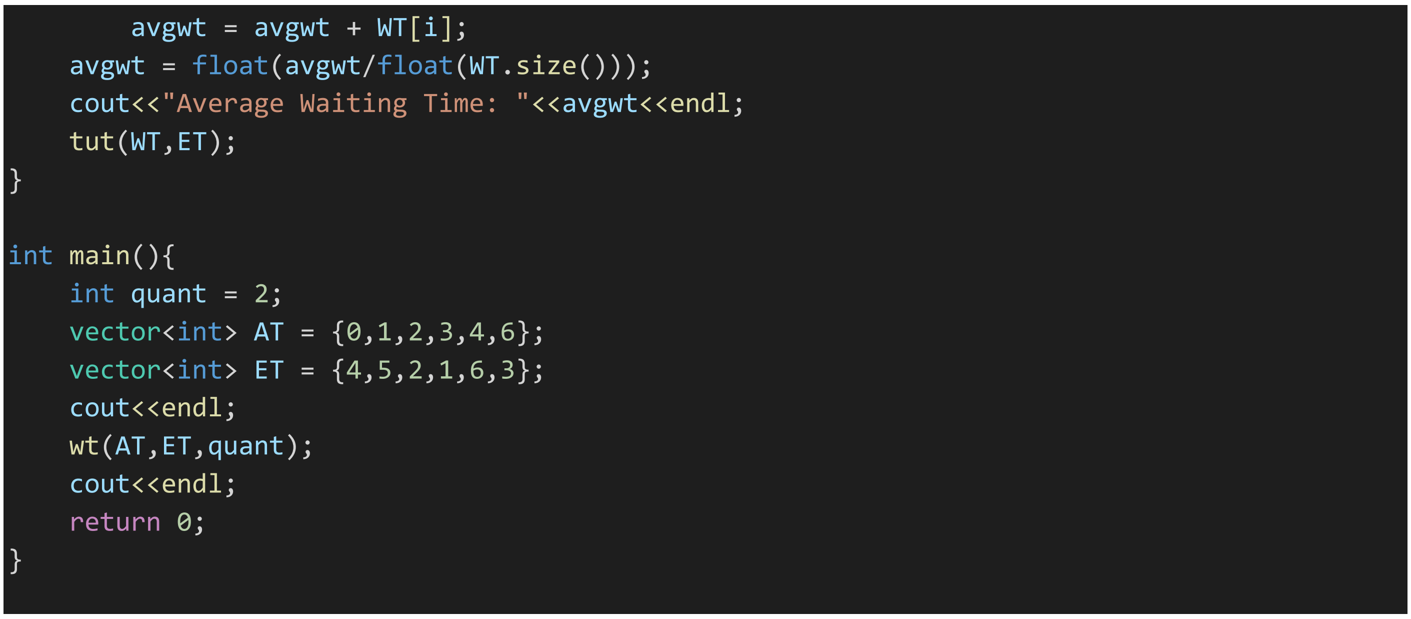
Experiment IV

Aim:

Implement Round Robin Scheduling Algorithm (with any time quantum of your choice).

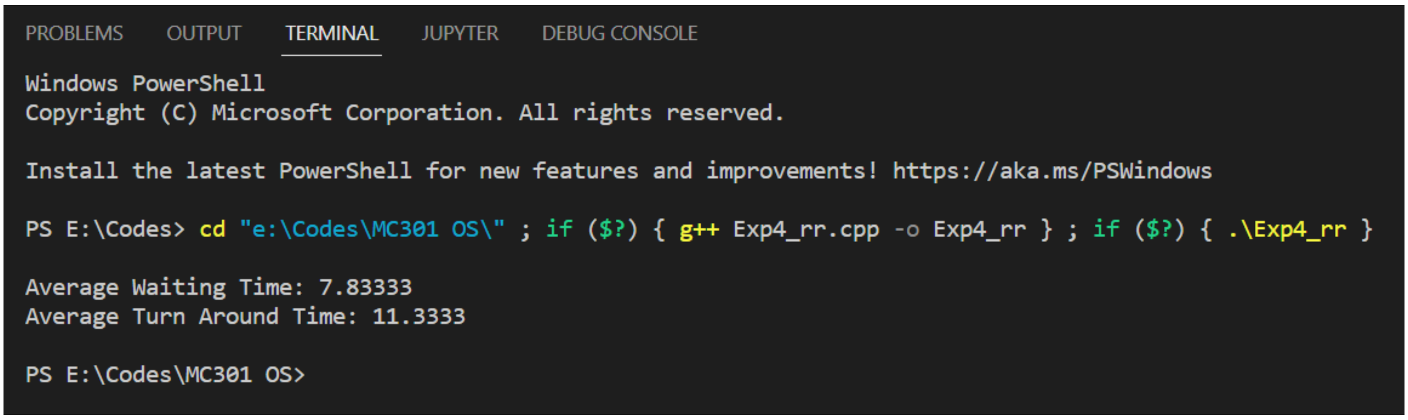
Code:

```
//Aneesh Panchal
//2K20/MC/21

#include<iostream>
#include<vector>
using namespace std;

void tut(vector<int> WT, vector<int> ET){
    vector<int> TUT(WT.size(),0);
    float avgtut = 0;
    for(int i=0;i<WT.size();i++){
        TUT[i]=WT[i]+ET[i];
        avgtut = avgtut + TUT[i];
    }
    avgtut = float(avgtut/float(WT.size()));
    cout<<"Average Turn Around Time: "<<avgtut<<endl;
}

void wt(vector<int> AT, vector<int> ET,int quant){
    vector<int> BT = ET;
    vector<int> WT(AT.size(),0);
    int t = AT[0];
    while(true){
        bool flag = true;
        for(int i=0;i<AT.size();++i)
            if(BT[i]>0){
                flag = false;
                if(BT[i]>quant){
                    t = t + quant;
                    BT[i] = BT[i] - quant;
                    AT[i] = AT[i] + quant;
                }
                else{
                    t = t + BT[i];
                    WT[i] = t - AT[i] - BT[i];
                    BT[i] = 0;
                }
            }
        if(flag)
            break;
    }
    float avgwt = 0;
    for(int i=0;i<AT.size();i++)
```



```cpp
        avgwt = avgwt + WT[i];
    avgwt = float(avgwt/float(WT.size()));
    cout<<"Average Waiting Time: "<<avgwt<<endl;
    tut(WT,ET);
}

int main(){
    int quant = 2;
    vector<int> AT = {0,1,2,3,4,6};
    vector<int> ET = {4,5,2,1,6,3};
    cout<<endl;
    wt(AT,ET,quant);
    cout<<endl;
    return 0;
}
```

Output:



```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS E:\Codes> cd "e:\Codes\MC301 OS\" ; if ($?) { g++ Exp4_rr.cpp -o Exp4_rr } ; if ($?) { .\Exp4_rr }

Average Waiting Time: 7.83333
Average Turn Around Time: 11.3333

PS E:\Codes\MC301 OS>
```

Aneesh Panchal
2K20/MC/21

Experiment V

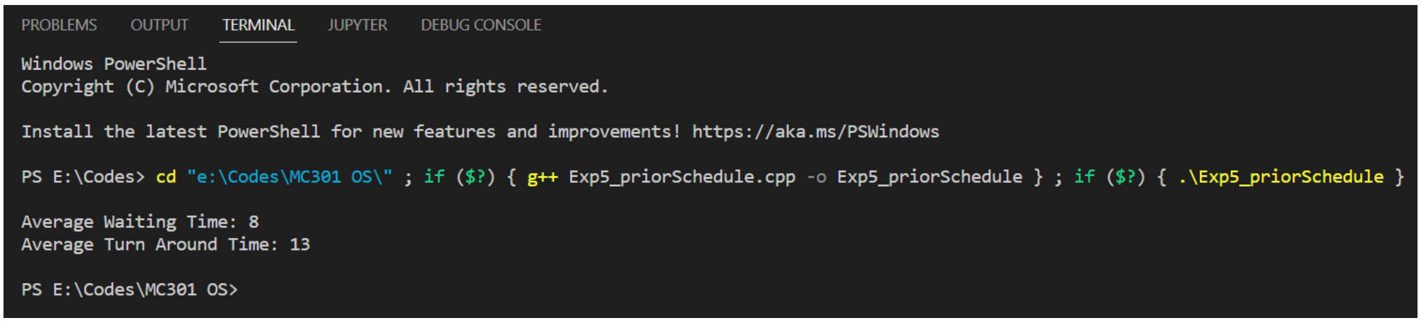
Aim:

Implement Priority based Job scheduling algorithm (non-preemptive version).

Code:

```cpp
//Aneesh Panchal
//2K20/MC/21

#include<iostream>
#include<vector>
using namespace std;

int priority(vector<bool> done, vector<int> AT, vector<int> PT, int t){
    int p = -1;
    for(int i=0;i<AT.size();++i){
        if(p==-1 && done[i]==false)
            p=i;
        else if((AT[i]<=t) && (done[i]==false) && (PT[i]<PT[p]))
            p=i;
    }
    return p;
}

void tut(vector<int> WT, vector<int> ET){
    vector<int> TUT(WT.size(),0);
    float avgtut = 0;
    for(int i=0;i<WT.size();i++){
        TUT[i]=WT[i]+ET[i];
        avgtut = avgtut + TUT[i];
    }
    avgtut = float(avgtut/float(WT.size()));
    cout<<"Average Turn Around Time: "<<avgtut<<endl;
}

void wt(vector<int> AT, vector<int> ET, vector<int> PT){
    vector<int> WT(AT.size(),0);
    vector<bool> done(AT.size(),false);
    int t = AT[0];
    for(int i=0;i<AT.size();i++){
        int index = priority(done,AT,PT,t);
        WT[index] = t - AT[index];
        t = t + ET[index];
        done[index] = true;
    }
    float avgwt = 0;
    for(int i=0;i<AT.size();i++)
        avgwt = avgwt + WT[i];
    avgwt = float(avgwt/float(WT.size()));
    cout<<"Average Waiting Time: "<<avgwt<<endl;
    tut(WT,ET);
}

int main(){
    vector<int> AT = {1,4,4,6};
    vector<int> ET = {10,3,5,2};
    vector<int> PT = {2,3,1,4};
    cout<<endl;
    wt(AT,ET,PT);
    cout<<endl;
    return 0;
}
```

Output:



```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS E:\Codes> cd "e:\Codes\MC301 OS\" ; if ($?) { g++ Exp5_priorSchedule.cpp -o Exp5_priorSchedule } ; if ($?) { .\Exp5_priorSchedule }

Average Waiting Time: 8
Average Turn Around Time: 13

PS E:\Codes\MC301 OS>
```

Aneesh Panchal
2K20/MC/21

Experiment VI

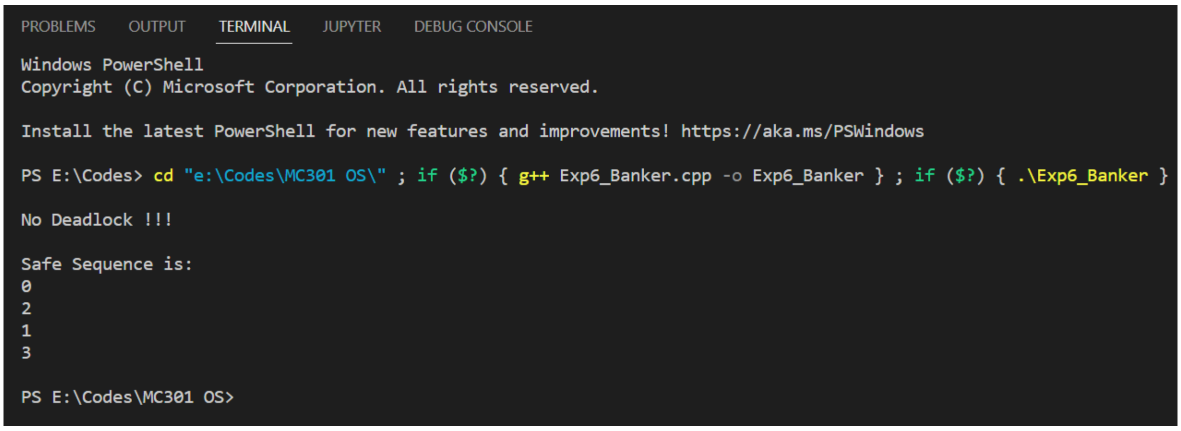
Aim:

Write a program to implement Banker's Algorithm for Deadlock Avoidance.

Code:

```cpp
//Aneesh Panchal
//2K20/MC/21

#include<iostream>
#include<vector>
using namespace std;

int kval(vector<vector<int>> req, vector<int> available, int i){
    int k = 0;
    for(int m=0;m<available.size();++m)
        if(req[i][m]<=available[m])
            ++k;
    return k;
}

bool deadlock(vector<vector<int>> req, vector<int> available, vector<int> done){
    bool dead = true;
    int k=0;
    for(int i=0;i<req.size();++i){
        k = kval(req,available,i);
        if(done[i]==false && k==available.size()){
            dead = false;
            break;
        }
        else
            continue;
    }
    return dead;
}

vector<int> allocation(vector<vector<int>> alloc,vector<int> available, int i){
    for(int m=0;m<available.size();++m)
        available[m] = alloc[i][m] + available[m];
    return available;
}

vector<int> safeseq(vector<vector<int>> alloc, vector<vector<int>> req, vector<int> available){
    vector<int> safe(req.size(),0);
    vector<int> done(req.size(),false);
    int j=0;
    for(int i=0;i<req.size();++i){
        int k=0;
        k = kval(req,available,i);
        if(done[i]==false && k==available.size()){
            safe[j] = i;
            done[i] = true;
            available = allocation(alloc,available,i);
            ++j;
        }
        if(j==req.size())
            break;
        if(i==req.size()-1){
            bool dead = deadlock(req,available,done);
            if(dead){
                safe[0] = -1;
                break;
            }
            i=0;
        }
    }
    return safe;
}

int main(){
    vector<int> available = {3,3,0};
    vector<vector<int>> alloc = {{1,0,1},{1,1,2},{1,0,3},{2,0,0}};
    vector<vector<int>> maximum = {{4,3,1},{2,1,4},{1,3,3},{5,4,1}};
    vector<vector<int>> req(alloc.size(),vector<int>(alloc[0].size(),0));
    for(int i=0;i<alloc.size();++i)
        for(int j=0;j<alloc[0].size();++j)
            req[i][j] = maximum[i][j] - alloc[i][j];
    vector<int> safe = safeseq(alloc,req,available);
    if(safe[0]==-1)
        cout<<endl<<"Deadlock Exists !!!"<<endl;
    else{
        cout<<endl<<"No Deadlock !!!"<<endl<<endl<<"Safe Sequence is:"<<endl;
        for(int i=0;i<safe.size();++i)
            cout<<safe[i]<<endl;
        cout<<endl;
    }
    return 0;
}
```

Output:



```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS E:\Codes> cd "e:\Codes\MC301 OS\" ; if ($?) { g++ Exp6_Banker.cpp -o Exp6_Banker } ; if ($?) { .\Exp6_Banker }

No Deadlock !!!

Safe Sequence is:
0
2
1
3

PS E:\Codes\MC301 OS>
```

Aneesh Panchal
2K20/MC/21

Experiment VII

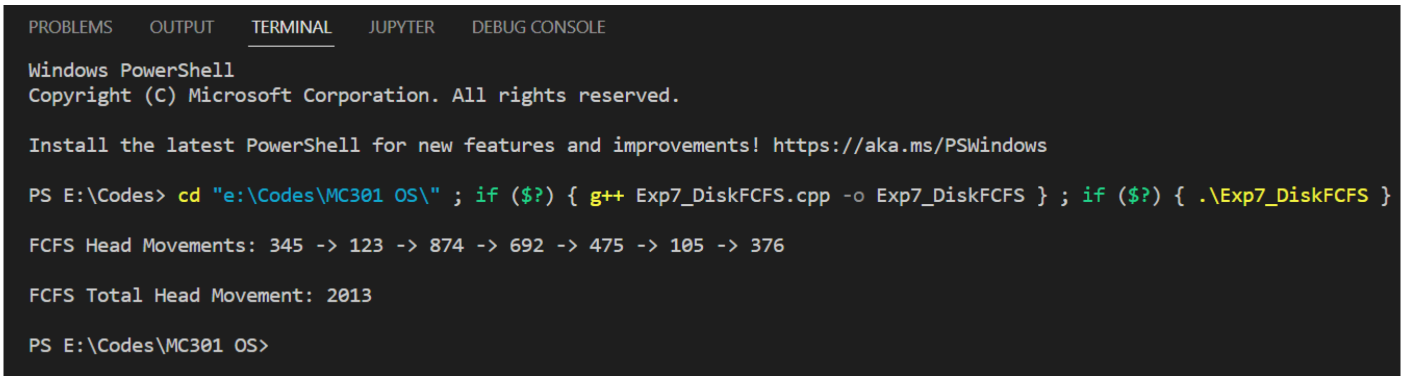
Aim:

Write a program to implement FCFS Disk Scheduling Algorithm.

Code:

```cpp
//Aneesh Panchal
//2K20/MC/21

#include<iostream>
#include<vector>
using namespace std;

int total_head_movement(vector<int> Track, int head){
    int total = 0;
    cout<<head;
    for(int i=0;i<Track.size();i++){
        cout<<" -> "<<Track[i];
        total = total + abs(head-Track[i]);
        head = Track[i];
    }
    cout<<endl<<endl;
    return total;
}

int main(){
    int head = 345;
    vector<int> Track = {123, 874, 692, 475, 105, 376};
    cout<<endl;
    cout<<"FCFS Head Movements: ";
    cout<<"FCFS Total Head Movement: "<<total_head_movement(Track,head)<<endl;
    cout<<endl;
    return 0;
}
```

Output:



```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS E:\Codes> cd "e:\Codes\MC301 OS\" ; if ($?) { g++ Exp7_DiskFCFS.cpp -o Exp7_DiskFCFS } ; if ($?) { .\Exp7_DiskFCFS }

FCFS Head Movements: 345 -> 123 -> 874 -> 692 -> 475 -> 105 -> 376

FCFS Total Head Movement: 2013

PS E:\Codes\MC301 OS>
```

Aneesh Panchal
2K20/MC/21

Experiment VIII

Aim:

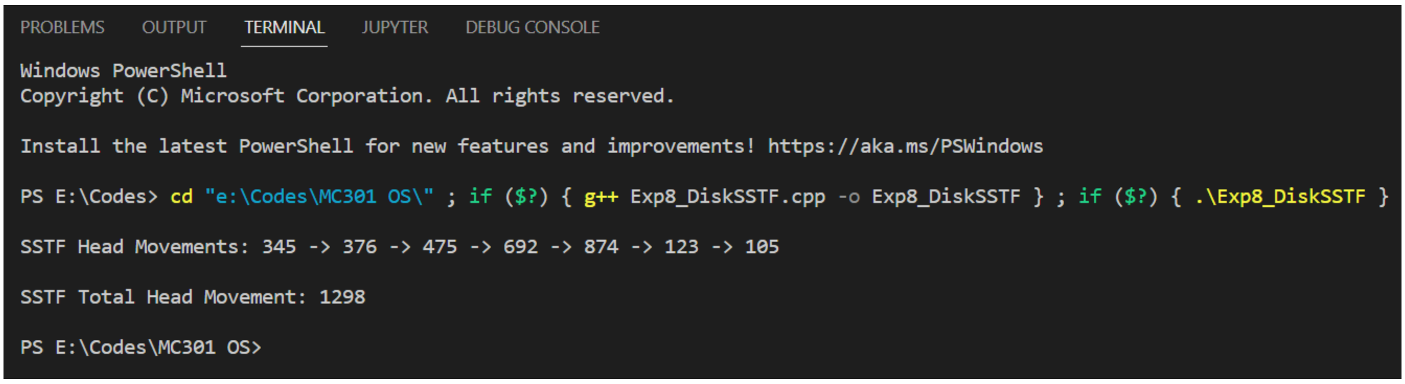
Write a program to implement SSTF Disk Scheduling Algorithm.

Code:

```
//Aneesh Panchal
//2K20/MC/21

#include<iostream>
#include<vector>
using namespace std;

vector<int> SSTF(vector<int> Track, int head){
    vector<int> SSTF(Track.size(),0);
    vector<bool> done(Track.size(),false);
    for(int i=0;i<Track.size();i++){
        int min = INT16_MAX;
        int index = 0;
        for(int j=0;j<Track.size();j++)
            if(done[j]==false && abs(head-Track[j])<min){
                min = abs(head-Track[j]);
                index = j;
            }
        SSTF[i] = Track[index];
        done[index] = true;
        head = Track[index];
    }
    return SSTF;
}

int total_head_movement(vector<int> Track, int head){
    int total = 0;
    for(int i=0;i<Track.size();i++){
        total = total + abs(head-Track[i]);
        head = Track[i];
    }
    cout<<endl<<endl;
    return total;
}

int main(){
    int head = 345;
    vector<int> Track = {123, 874, 692, 475, 105, 376};
    vector<int> SSTF_track = SSTF(Track,head);
    cout<<endl;
    cout<<"SSTF Head Movements: ";
    cout<<head;
    for(int i=0;i<SSTF_track.size();i++)
```

```
        cout<<" -> "<<SSTF_track[i];
    cout<<"SSTF Total Head Movement: "<<total_head_movement(SSTF_track,head)<<endl;
    cout<<endl;
    return 0;
}
```

Output:



```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS E:\Codes> cd "e:\Codes\MC301 OS\" ; if ($?) { g++ Exp8_DiskSSTF.cpp -o Exp8_DiskSSTF } ; if ($?) { .\Exp8_DiskSSTF }

SSTF Head Movements: 345 -> 376 -> 475 -> 692 -> 874 -> 123 -> 105

SSTF Total Head Movement: 1298

PS E:\Codes\MC301 OS>
```