TABLE OF CONTENT

LAB 1 [SIMULATION OF FCFS AND SJF SCHEDULING ALGORITHM]

Given the list of processes, their CPU burst time. WAP to simulate FCFS and SJF algorithm. The program should display the Gantt Chart and compute the average waiting time and average turnaround time for each of the scheduling algorithms.

FCFS:

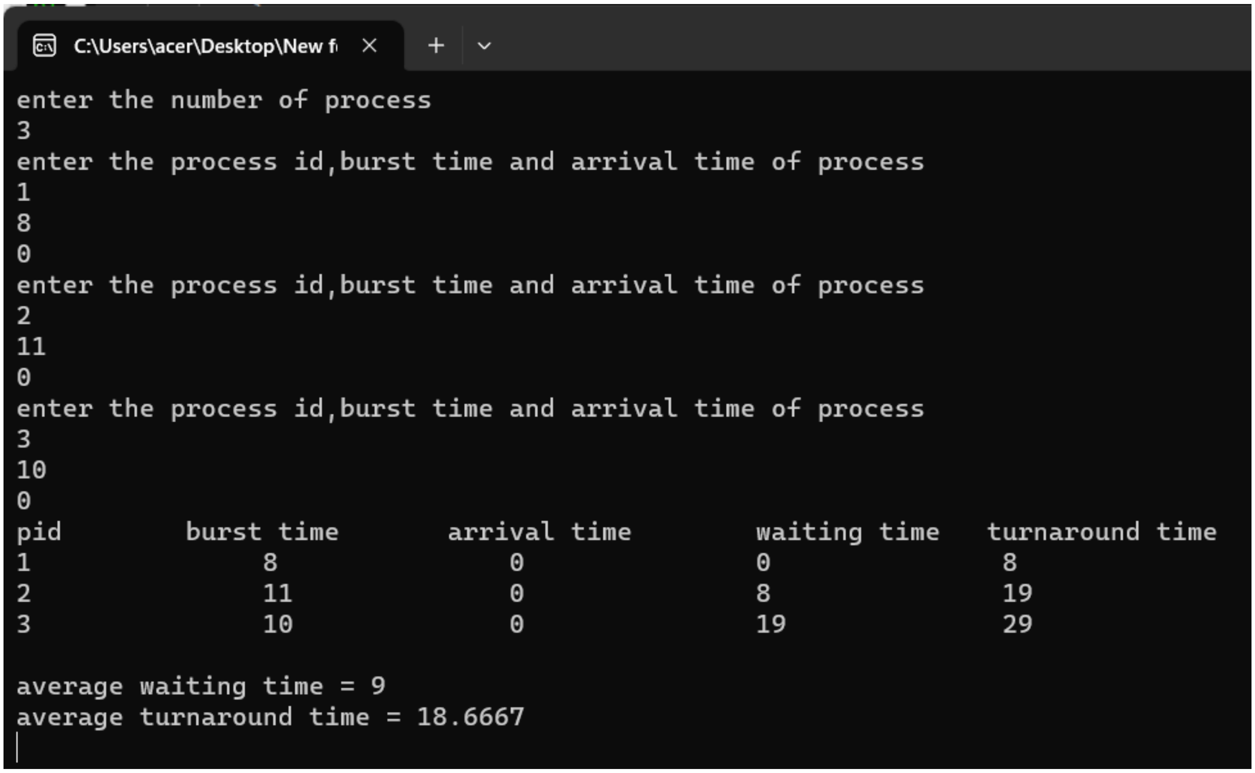
Program:

```
#include<iostream>
#include<conio.h>
#include<iomanip>
using namespace std;
float avgwt,avgtt;
int n;
class Process
{
	int id,bt,at,wt,tt,ft;
	public:
		void getdata()
		{
			cout<<"enter the process id,burst time and arrival time of process"<<endl;
			cin>>id>>bt>>at;
		}
		void showdata()
		{
			cout<<id<<"\t\t"<<bt<<"\t\t"<<at<<"\t\t  "<<wt<<"\t\t  "<<tt<<endl;
		}
		void sorting(Process p[])
		{
			Process temp;
			for(int i=0;i<n-1;i++)
			{
				for(int j=0;j<n-1-i;j++)
				{
					if(p[j].at>p[j+1].at)
					{
						temp = p[j];
						p[j]=p[j+1];
						p[j+1]=temp;
					}
				}
			}
		 }
```

```
        void scheduler(Process p[])
         {
            int time = p[0].at;
            for(int i=0;i<n;i++)
            {
                if(i==0)
                {
                    p[i].wt = 0;
                    p[i].tt = p[i].bt;
                    p[i].ft = p[i].bt;

                }
                else
                {
                    if(time >= p[i].at)
                    {
                        p[i].ft = p[i-1].ft + p[i].bt;
                        p[i].wt = p[i-1].ft-p[i].at;
                        p[i].tt = p[i].wt+p[i].bt;
                    }
                }
                time = time+p[i].bt;
            }
            for(int i=0;i<n;i++)
            {
                 avgwt=avgwt+p[i].wt;
                avgtt = avgtt+p[i].tt;
            }
         }
};

int main()
{
    Process p[10],S;
    cout<<"enter the number of process"<<endl;
    cin>>n;
    for(int i=0;i<n;i++)
    {
        p[i].getdata();
    }
    S.sorting(p);
    S.scheduler(p);
    cout<<"pid        "<<"burst time       "<<"arrival time        "<<"waiting time "<<
        "turnaround time"<<endl;
    for(int i=0;i<n;i++)
    {
        p[i].showdata();
    }
```

```
        cout<<"\naverage waiting time = "<<avgwt/n<<endl;
        cout<<"average turnaround time = "<<avgtt/n<<endl;
        getch();
        return 0;
}
```

OUTPUT:



```
enter the number of process
3
enter the process id,burst time and arrival time of process
1
8
0
enter the process id,burst time and arrival time of process
2
11
0
enter the process id,burst time and arrival time of process
3
10
0
pid        burst time       arrival time        waiting time   turnaround time
1              8               0                0              8
2              11              0                8              19
3              10              0                19             29

average waiting time = 9
average turnaround time = 18.6667
```

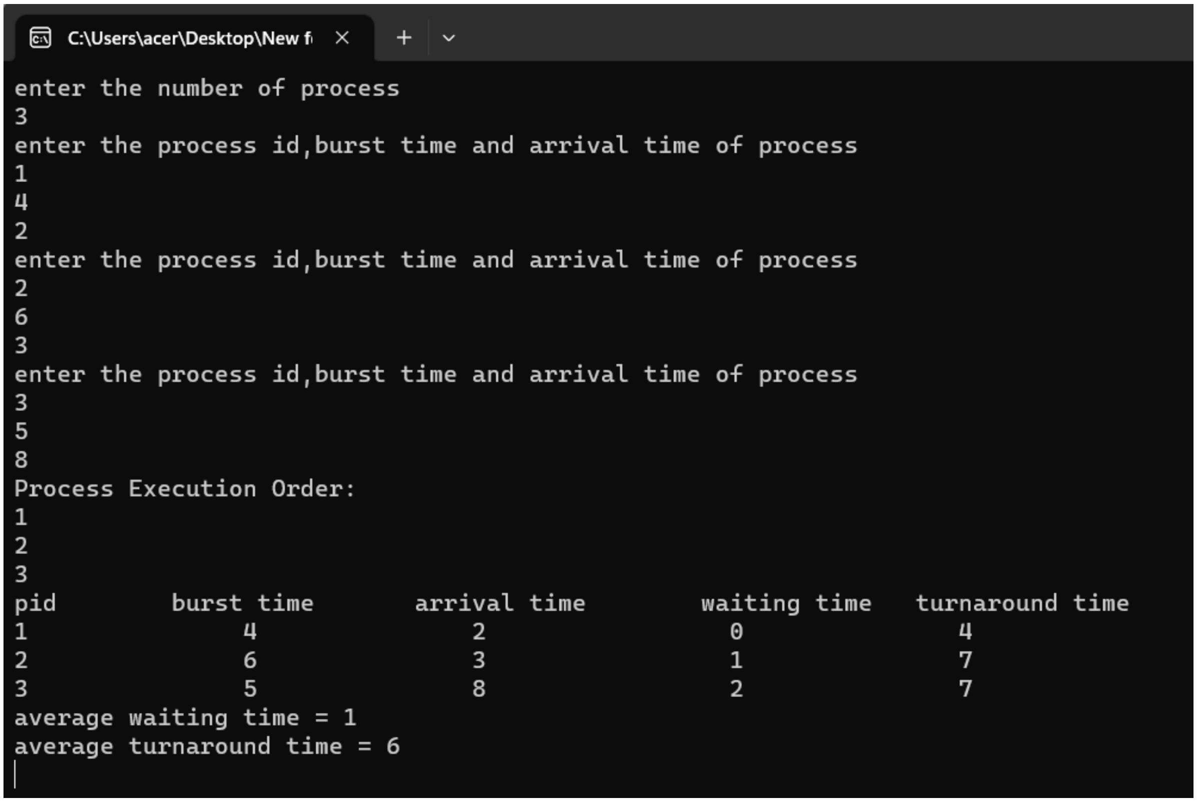
SJF

PROGRAM:

```
#include<iostream>
#include<conio.h>
#include<iomanip>
using namespace std;
float avgwt,avgtt;
int n;
class Process
{
    int id,bt,at,wt,tt,ft,flag;
    public:
        void getdata()
        {
            cout<<"enter the process id,burst time and arrival time of process"<<endl;
            cin>>id>>bt>>at;
        }
        void showdata()
        {
            cout<<id<<"\t\t"<<bt<<"\t\t"<<at<<"\t\t  "<<wt<<"\t\t  "<<tt<<endl;
        }
        void sorting(Process p[])
        {
            Process temp;
            for(int i=0;i<n-1;i++)
            {
                for(int j=0;j<n-1-i;j++)
                {
                    if(p[j].at>p[j+1].at)
                    {
                        temp=p[j];
                        p[j]=p[j+1];
                        p[j+1]=temp;
                    }
                    if(p[j].at==p[j+1].at)
                    {
                        if(p[j].bt>p[j+1].bt)
                        {
                            temp=p[j];
                            p[j]=p[j+1];
                            p[j+1]=temp;
                        }
                    }
                }
            }
        }
        void scheduler(Process p[])
        {
            Process temp;
            int time=p[0].at,min=999,k;
            for(int j=0;j<n;j++)
            {
                if(j==0)
                {
                    p[j].wt=0;
                    p[j].tt=p[j].bt;
                    p[j].ft=p[j].bt;
                    temp=p[j];
                    time=time+p[j].bt;
                    cout<<p[j].id<<" "<<endl;
                }
                else
                {
                    for(int i=1;i<n;i++)
                    {
                        if(time>=p[i].at && min>p[i].bt && p[i].flag!=1)
                        {
                            min=p[i].bt;
                            k=i;
                        }
                    }
                    cout<<p[k].id<<" "<<endl;
                    time=time+min;
                    p[k].flag=1;
                    p[k].ft= min +temp.ft;
                    p[k].wt=temp.ft-p[k].at;
                    p[k].tt=p[k].wt+p[k].bt;
                    temp=p[k];
                    min=999;
                }
             }
            for(int i=0;i<n;i++)
            {
                 avgwt=avgwt+p[i].wt;
                avgtt = avgtt+p[i].tt;
            }
        }
};
```

```
int main()
{
        Process p[10],S;
        cout<<"enter the number of process"<<endl;
        cin>>n;
        for(int i=0;i<n;i++)
        {
                p[i].getdata();
        }
        cout<<"Process Execution Order:"<<endl;
        S.sorting(p);
        S.scheduler(p);
        cout<<"pid      "<<"burst time      "<<"arrival time       "<<"waiting time   "<<
                "turnaround time"<<endl;
        for(int i=0;i<n;i++)
        {
                p[i].showdata();
        }
        cout<<"average waiting time = "<<avgwt/n<<endl;
        cout<<"average turnaround time = "<<avgtt/n<<endl;
        getch();
        return 0;
}
```

OUTPUT:



```
enter the number of process
3
enter the process id,burst time and arrival time of process
1
4
2
enter the process id,burst time and arrival time of process
2
6
3
enter the process id,burst time and arrival time of process
3
5
8
Process Execution Order:
1
2
3
pid          burst time        arrival time          waiting time    turnaround time
1                 4                 2                   0                4
2                 6                 3                   1                7
3                 5                 8                   2                7
average waiting time = 1
average turnaround time = 6
```

LAB 2 [SIMULATION OF RR AND PRIORITY SCHEDULING ALGORITHM]

Given the list of processes, their CPU burst time. WAP to simulate RR and Priority algorithm. The program should display the Gantt Chart and compute the average waiting time and average turnaround time for each of the scheduling algorithms.

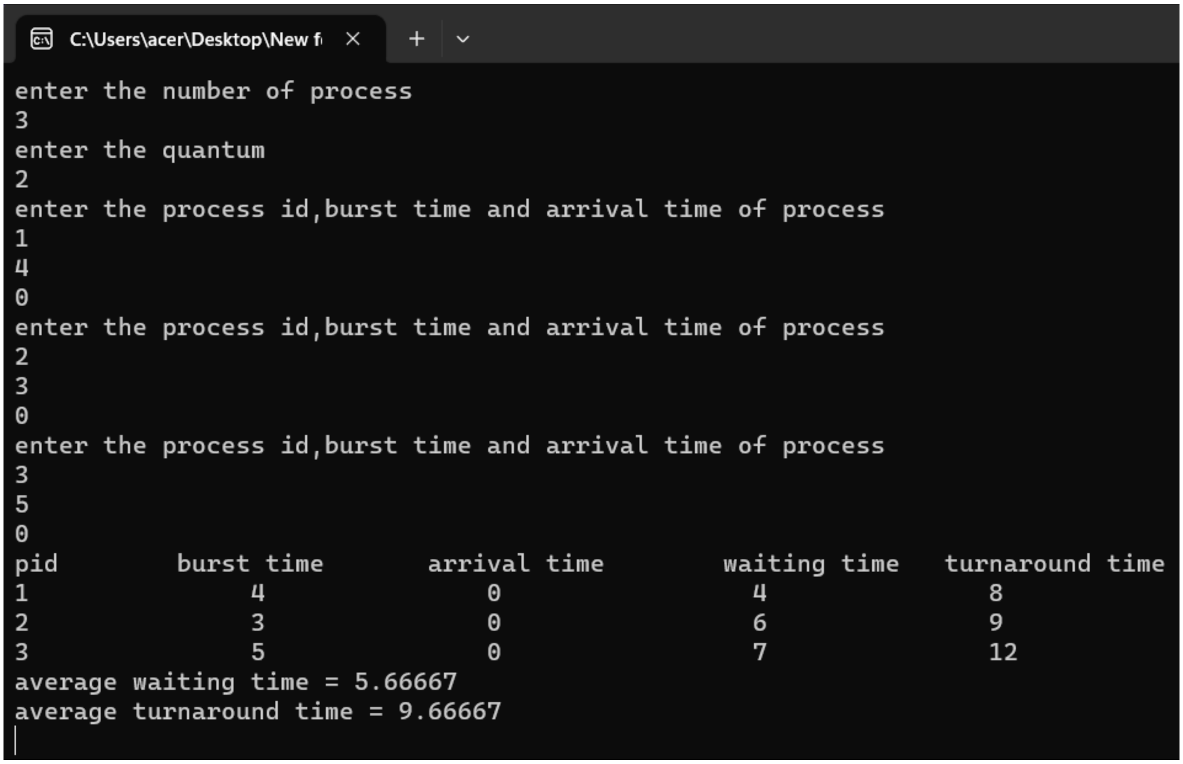
RR:

PROGRAM:

```
#include<iostream>
#include<conio.h>
using namespace std;
float avgwt,avgtt;
int n,quantum;
class Process
{
    int id,bt,at,wt,tt,ft;
    int dup[10];
    public:
        void getdata()
        {
            cout<<"enter the process id,burst time and arrival time of process"<<endl;
            cin>>id>>bt>>at;
        }
        void showdata()
        {
            cout<<id<<"\t\t"<<bt<<"\t\t"<<at<<"\t\t  "<<wt<<"\t\t  "<<tt<<endl;
        }
        void sorting(Process p[])
        {
            Process temp;
            for(int i=0;i<n-1;i++)
            {
                for(int j=0;j<n-1-i;j++)
                {
                    if(p[j].at>p[j+1].at)
                    {
                        temp=p[j];
                        p[j]=p[j+1];
                        p[j+1]=temp;
                    }
                }
            }
        }
        void scheduler(Process p[])
        {
            Process temp;
            int total=0,time=p[0].at;
            for(int i=0;i<n;i++)
            {
                total = total+p[i].bt;
                dup[i]=p[i].bt;
            }
            do
            {
                for(int i=0;i<n;i++)
                {
                    if(time>=p[i].at)
                    {
                        if(dup[i]>quantum)
                        {
                            time= time+ quantum;
                            p[i].ft=time;
                            dup[i]=dup[i]-quantum;
                        }
                        else
                        {
                            if(dup[i]!=0)
                            {
                                time = time + dup[i];
                                p[i].ft=time;
                                dup[i]=0;
                            }
                        }
                    }
                }

            }while(time!=total);
            for(int i=0;i<n;i++)
            {
                p[i].wt = p[i].ft - p[i].bt-p[i].at;
                p[i].tt = p[i].wt + p[i].bt;
                avgwt=avgwt+p[i].wt;
                avgtt = avgtt+p[i].tt;
            }
        }
};
```

```
int main()
{
        Process p[10],S;
        cout<<"enter the number of process"<<endl;
        cin>>n;
        cout<<"enter the quantum"<<endl;
        cin>>quantum;
        for(int i=0;i<n;i++)
        {
                p[i].getdata();
        }
        S.sorting(p);
        S.scheduler(p);
        cout<<"pid      "<<"burst time      "<<"arrival time      "<<"waiting time   "<<"turnaround
                time"<<endl;
        for(int i=0;i<n;i++)
        {
                p[i].showdata();
        }
        cout<<"average waiting time = "<<avgwt/n<<endl;
        cout<<"average turnaround time = "<<avgtt/n<<endl;
        getch();
        return 0;
}
```

OUTPUT:



```
enter the number of process
3
enter the quantum
2
enter the process id,burst time and arrival time of process
1
4
0
enter the process id,burst time and arrival time of process
2
3
0
enter the process id,burst time and arrival time of process
3
5
0
pid        burst time       arrival time        waiting time   turnaround time
1               4               0                 4              8
2               3               0                 6              9
3               5               0                 7              12
average waiting time = 5.66667
average turnaround time = 9.66667
```

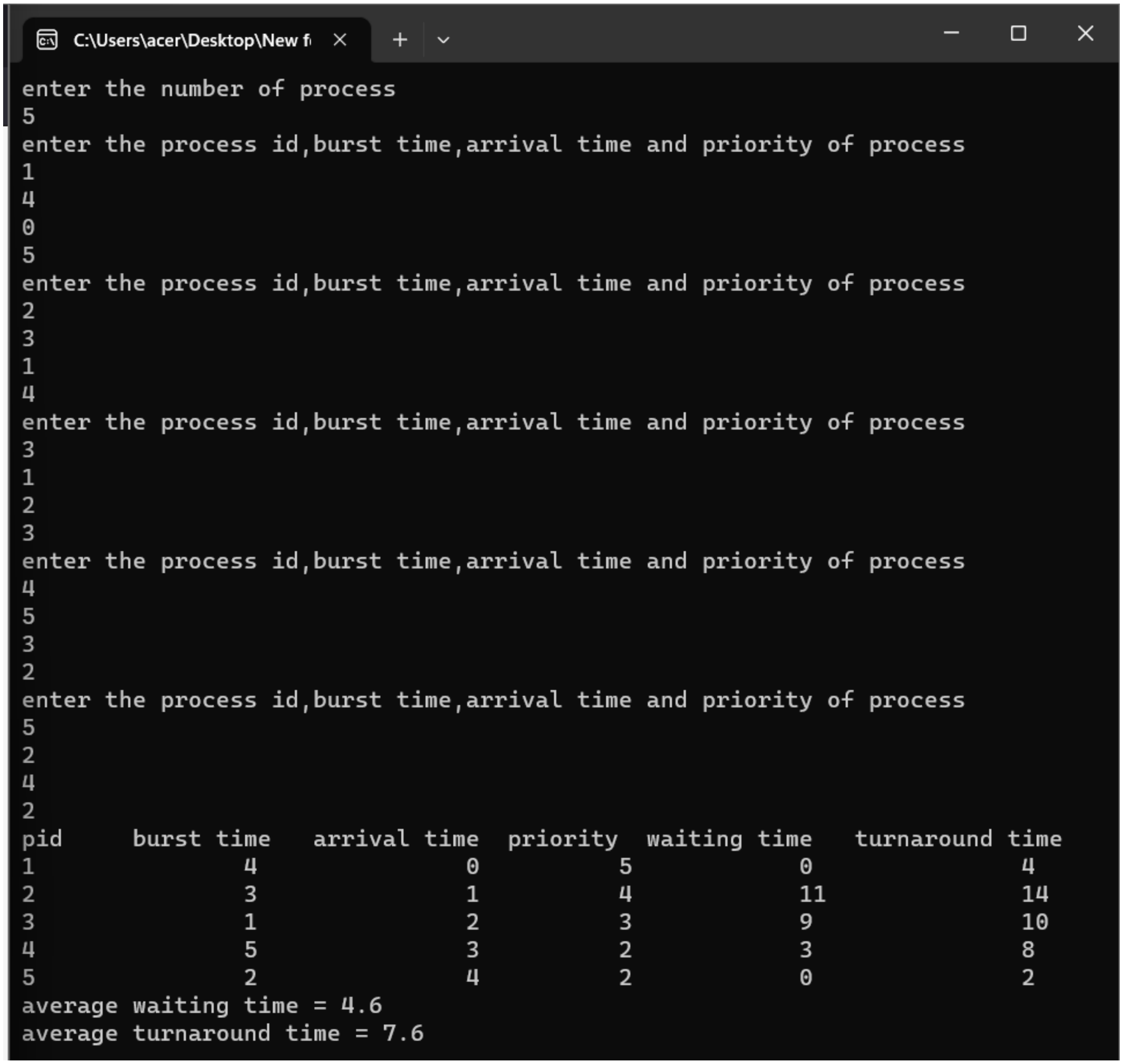
PRIORITY:

PROGRAM:

```
#include<iostream>
#include<conio.h>
using namespace std;
float avgwt,avgtt;
int n;
class Process
{
    int id,bt,at,wt,tt,ft,pr,flag;
    public:
        void getdata()
        {
            cout<<"enter the process id,burst time,arrival time and priority of
                process"<<endl;
            cin>>id>>bt>>at>>pr;
        }
        void showdata()
        {
            cout<<id<<"\t\t"<<bt<<"\t\t"<<at<<"\t   "<<pr<<
                "\t\t"<<wt<<"\t\t"<<tt<<endl;
        }
        void sorting(Process p[])
        {
            Process temp;
            for(int i=0;i<n-1;i++)
            {
                for(int j=0;j<n-1-i;j++)
                {
                    if(p[j].at>p[j+1].at)
                    {
                        temp=p[j];
                        p[j]=p[j+1];
                        p[j+1]=temp;
                    }
                    if(p[j].at==p[j+1].at)
                    {
                        if(p[j].pr>p[j+1].pr)
                        {
                            temp=p[j];
                            p[j]=p[j+1];
                            p[j+1]=temp;
                        }
                    }
                }
            }
        }
        void scheduler(Process p[])
        {
            Process temp;
            int time=p[0].at,k=0,priority=999;
            for(int j=0;j<n;j++)
            {
                if(j==0)
                {
                    p[j].wt=0;
                    p[j].tt=p[j].bt;
                    p[j].ft=p[j].bt;
                    temp=p[j];
                    time=time+p[j].bt;
                }
                else
                {
                    for(int i=1;i<n;i++)
                    {
                       if(time>=p[i].at && priority>=p[i].pr && p[i].flag!=1)
                        {
                            priority = p[i].pr;
                            k=i;
                        }
                    }
                    time=time+p[k].bt;
                    p[k].flag=1;
                    p[k].ft= p[k].bt +temp.ft;
                    p[k].wt=temp.ft-p[k].at;
                    p[k].tt=p[k].wt+p[k].bt;
                    temp=p[k];
                    priority=999;
                }

            }
            for(int i=0;i<n;i++)
            {
              avgwt=avgwt+p[i].wt;
                 avgtt = avgtt+p[i].tt;
            }
        }

};
```

```
int main()
{
        Process p[100],S;
        cout<<"enter the number of process"<<endl;
        cin>>n;
        for(int i=0;i<n;i++)
        {
                p[i].getdata();
        }
        S.sorting(p);
        S.scheduler(p);
         cout<<"pid     "<<"burst time   "<<"arrival time  "<<"priority  "<<"waiting time   "<<
                "turnaround time"<<endl;
        for(int i=0;i<n;i++)
        {
                p[i].showdata();
        }
        cout<<"average waiting time = "<<avgwt/n<<endl;
        cout<<"average turnaround time = "<<avgtt/n<<endl;
        getch();
        return 0;
}
```

OUTPUT:



```
enter the number of process
5
enter the process id,burst time,arrival time and priority of process
1
4
0
5
enter the process id,burst time,arrival time and priority of process
2
3
1
4
enter the process id,burst time,arrival time and priority of process
3
1
2
3
enter the process id,burst time,arrival time and priority of process
4
5
3
2
enter the process id,burst time,arrival time and priority of process
5
2
4
2
pid     burst time   arrival time  priority  waiting time   turnaround time
1               4               0          5            0               4
2               3               1          4            11              14
3               1               2          3            9               10
4               5               3          2            3               8
5               2               4          2            0               2
average waiting time = 4.6
average turnaround time = 7.6
```

LAB 3 [SIMULATION OF PAGE REPLACEMENT ALGORITHM]

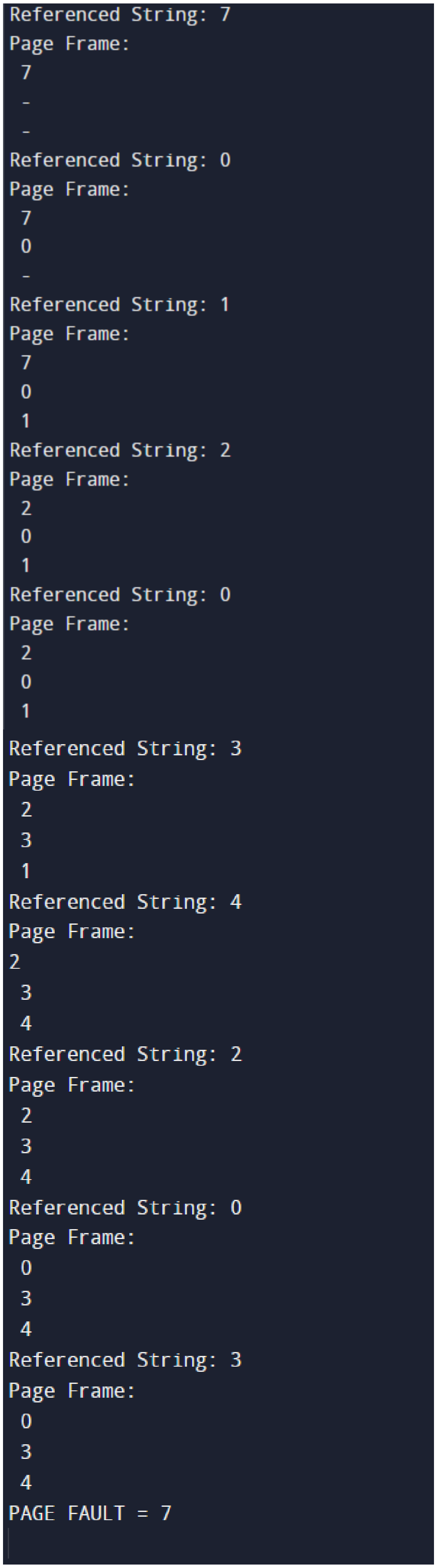
Given the list of referenced string and page frame. WAP to simulate FIFO, LRU and CLOCK replacement algorithm. The program should display the sequence and compute the total page fault.

FIFO:

PROGRAM:

```
#include<iostream>
#include<conio.h>
using namespace std;
int a[20]={7,0,1,2,0,3,0,4,2,3,0,3,2,1,2,0,1,7,0,1};
int pf[3]={-1,-1,-1};
int flag,nnext,pagefault;
int i,j,k;
int main()
{
        for(j=0;j<20;j++)
        {
                for(i=0;i<3;i++)
                {
                        if(a[j]==pf[i])
                        {
                                flag=0;
                                break;
                        }
                        else
                        {
                                flag=1;
                        }
                }
                if(flag==1)
                {
                        pagefault++;
                        pf[nnext]=a[j];
                        nnext++;
                }
                if(nnext==3)
                {
                        nnext=0;
                }
                cout<<"Referenced String: "<<a[j]<<endl;
                cout<<"Page Frame: "<<endl;
        for(k=0;k<3;k++)
        {
            if(pf[k]==-1)
            {
                cout<<" - ";
            }
            else
            {
                cout<<” ”<<pf[k];
            }
            cout<<endl;
        }

    }
    cout<<"PAGE FAULT = "<<pagefault<<endl;
    getch();
    return 0;
}
```

OUTPUT:



```
Referenced String: 7
Page Frame:
 7
 -
 -
Referenced String: 0
Page Frame:
 7
 0
 -
Referenced String: 1
Page Frame:
 7
 0
 1
Referenced String: 2
Page Frame:
 2
 0
 1
Referenced String: 0
Page Frame:
 2
 0
 1
Referenced String: 3
Page Frame:
 2
 3
 1
Referenced String: 4
Page Frame:
2
 3
 4
Referenced String: 2
Page Frame:
 2
 3
 4
Referenced String: 0
Page Frame:
 0
 3
 4
Referenced String: 3
Page Frame:
 0
 3
 4
PAGE FAULT = 7
```

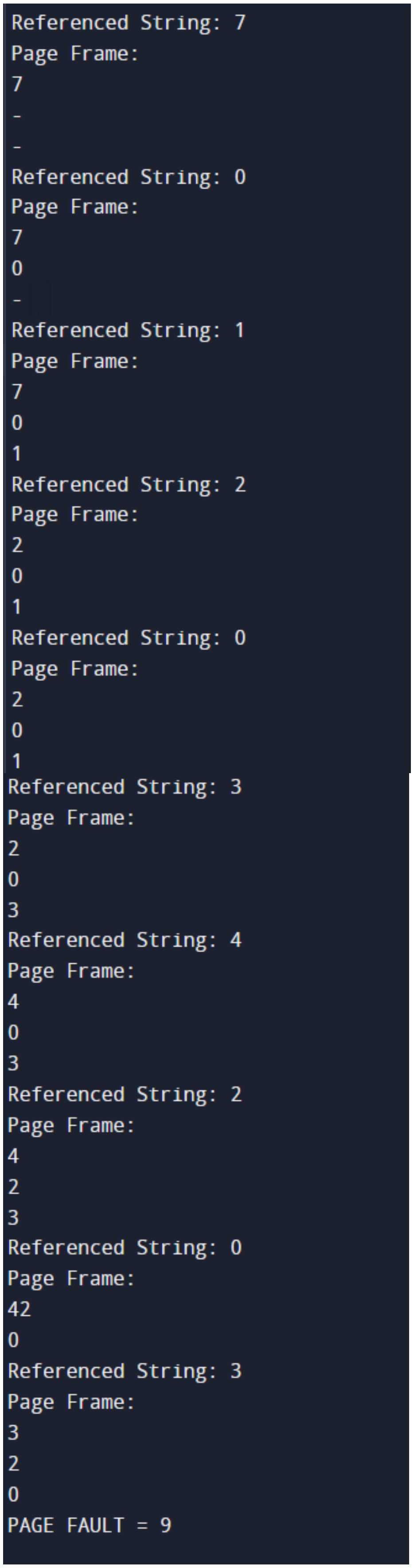
LRU:

PROGRAM:

```
#include<iostream>
#include<conio.h>
using namespace std;
int a[10]={7,0,1,2,0,3,4,2,0,3};
int pf[3]={-1,-1,-1};
int timme[3];
int nnext,pagefault,timer,minm,flag;
int i,j,k,m;
int main()
{
        for(j=0;j<10;j++)
        {
                for(i=0;i<3;i++)
                {
                        if(a[j]==pf[i])
                        {
                                flag=0;
                                nnext=i;
                                break;
                        }
                        else
                        {
                                flag=1;
                         }
                }
                if(flag==1)
                {
                        minm=999;
                        for(m=0;m<3;m++)
                        {
                                if(timme[m]<minm)
                                {
                                        minm=timme[m];
                                        nnext=m;
                                }
                        }
                        pagefault++;
                        pf[nnext]=a[j];
                        timer=timer+1;
                        timme[nnext]=timer;
                }
                else
                {
                        timer=timer+1;
                        timme[nnext]=timer;
                }
                cout<<"Referenced String: "<<a[j]<<endl;
                cout<<"Page Frame: "<<endl;
```

```
				for(k=0;k<3;k++)
				{
					if(pf[k]==-1)
					{
						cout<<"- "<<endl;
					}
					else
					{
						cout<<pf[k]<<endl;
					}
				}
			}
			cout<<"PAGE FAULT = "<<pagefault<<endl;
			getch();
			return 0;
		}
```

OUTPUT:



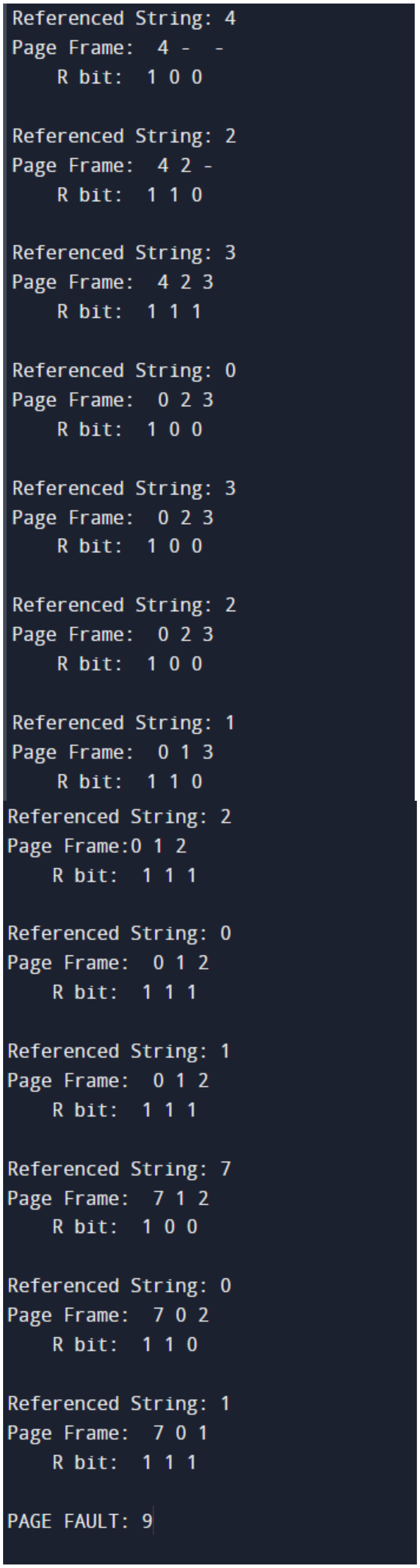
```
Referenced String: 7
Page Frame:
7
-
-
Referenced String: 0
Page Frame:
7
0
-
Referenced String: 1
Page Frame:
7
0
1
Referenced String: 2
Page Frame:
2
0
1
Referenced String: 0
Page Frame:
2
0
1
Referenced String: 3
Page Frame:
2
0
3
Referenced String: 4
Page Frame:
4
0
3
Referenced String: 2
Page Frame:
4
2
3
Referenced String: 0
Page Frame:
42
0
Referenced String: 3
Page Frame:
3
2
0
PAGE FAULT = 9
```

CLOCK:

PROGRAM:

```
#include<iostream>
#include<conio.h>
using namespace std;
int a[13]={4,2,3,0,3,2,1,2,0,1,7,0,1};
int pf[3]={-1,-1,-1};
int R[3];
int pagefault,flag,clockhand;
int i,j,k,m;
int main()
{
        for(j=0;j<13;j++)
        {
                for(i=0;i<3;i++)
                {
                        if(a[j]==pf[i])
                        {
                                flag=0;
                                break;
                        }
                        else
                        {
                                flag=1;
                        }
                }
                if(flag==1)
                {
                        for(m=clockhand;m<3;m++)
                        {
                                if(R[m]==0)
                                {
                                        clockhand=m;
                                        break;
                                }
                                else
                                {
                                        R[m]=0;
                                }
                        }
                        pf[clockhand]=a[j];
                        R[clockhand]=1;
                        clockhand++;
                        pagefault++;
                }
        if(clockhand==3)
        {
            clockhand=0;
        }
        cout<<"Referenced String: "<<a[j]<<endl;
        cout<<"Page Frame:\t";
        for(k=0;k<3;k++)
        {
            if(pf[k]==-1)
            {
                printf(" - ");
            }
            else
            {
                cout<<" "<<pf[k];
            }
            //cout<<endl;
        }
        cout<<endl<<"\tR bit:\t";
        for(m=0;m<3;m++)
        {
            cout<<R[m]<<" ";
        }
        cout<<endl<<endl;
    }
    cout<<"PAGE FAULT: "<<pagefault;
    getch();
    return 0;
}
```

OUTPUT:



```
Referenced String: 4
Page Frame:  4 -  -
    R bit:  1 0 0

Referenced String: 2
Page Frame:  4 2 -
    R bit:  1 1 0

Referenced String: 3
Page Frame:  4 2 3
    R bit:  1 1 1

Referenced String: 0
Page Frame:  0 2 3
    R bit:  1 0 0

Referenced String: 3
Page Frame:  0 2 3
    R bit:  1 0 0

Referenced String: 2
Page Frame:  0 2 3
    R bit:  1 0 0

Referenced String: 1
Page Frame:  0 1 3
    R bit:  1 1 0
Referenced String: 2
Page Frame:0 1 2
    R bit:  1 1 1

Referenced String: 0
Page Frame:  0 1 2
    R bit:  1 1 1

Referenced String: 1
Page Frame:  0 1 2
    R bit:  1 1 1

Referenced String: 7
Page Frame:  7 1 2
    R bit:  1 0 0

Referenced String: 0
Page Frame:  7 0 2
    R bit:  1 1 0

Referenced String: 1
Page Frame:  7 0 1
    R bit:  1 1 1

PAGE FAULT: 9
```

LAB 4 [SIMULATION OF DISK SCHEDULING ALGORITHM]

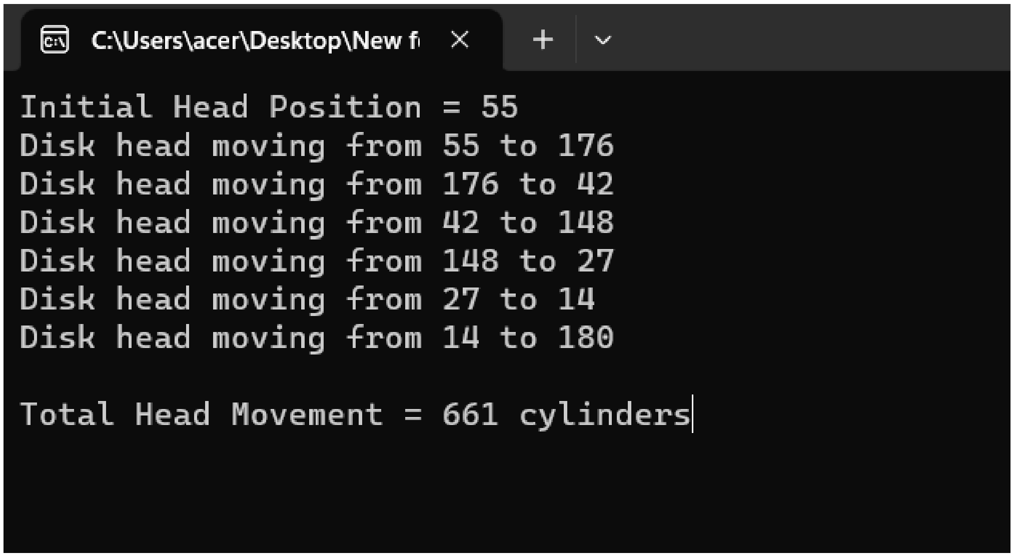
Given the list of disk cylinder and initial head position. WAP to simulate FCFS and SSTF disk scheduling algorithm. The program should display the sequence of head movement and compute the total head movement.

FCFS:

PROGRAM:

```
#include<iostream>
#include<conio.h>
using namespace std;
int d[7]={55,176,42,148,27,14,180};
int move=0,i;
int main()
{
        cout<<"Initial head position = "<<d[0]<<endl;
        for(i=1;i<7;i++)
        {
                cout<<"Disk head moving from "<<d[i-1]<<" to "<<d[i]<<endl;
                move = move + abs(d[i]-d[i-1]);
        }
        cout<<endl<<"Total head movement = "<<move<<" cylinders";
        getch();
        return 0;
}
```

OUTPUT:



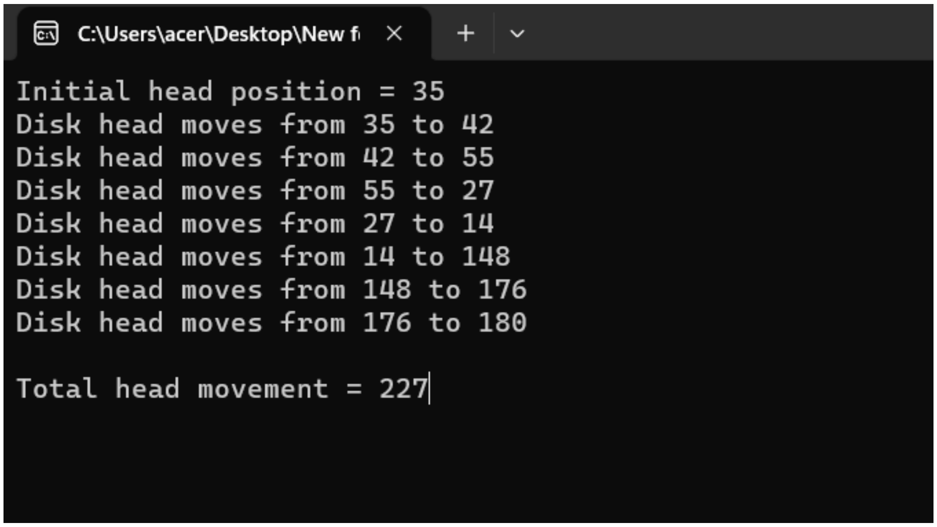
```
Initial Head Position = 55
Disk head moving from 55 to 176
Disk head moving from 176 to 42
Disk head moving from 42 to 148
Disk head moving from 148 to 27
Disk head moving from 27 to 14
Disk head moving from 14 to 180

Total Head Movement = 661 cylinders
```

SSTF:

PROGRAM:

```
#include<iostream>
#include<conio.h>
using namespace std;
int d[7]={55,176,42,148,27,14,180};
int diff[7], move=0, initial=35, minm=999;
int i,j,k;
int main()
{
        cout<<"Initial head position = "<<initial<<endl;
        for(j=0;j<7;j++)
        {
                for(i=0;i<7;i++)
                {
                        diff[i]= abs(initial-d[i]);
                        if(minm>diff[i]&&d[i]!=0)
                        {
                                minm=diff[i];
                                k=i;
                        }
                }
                cout<<"Disk head moves from "<<initial<<" to "<<d[k]<<endl;
                move= move + minm;
                initial = d[k];
                d[k]=0;
                minm=999;
        }
        cout<<endl<<"Total head movement = "<<move;
        getch();
        return 0;
}
```

OUTPUT:



```
Initial head position = 35
Disk head moves from 35 to 42
Disk head moves from 42 to 55
Disk head moves from 55 to 27
Disk head moves from 27 to 14
Disk head moves from 14 to 148
Disk head moves from 148 to 176
Disk head moves from 176 to 180

Total head movement = 227
```

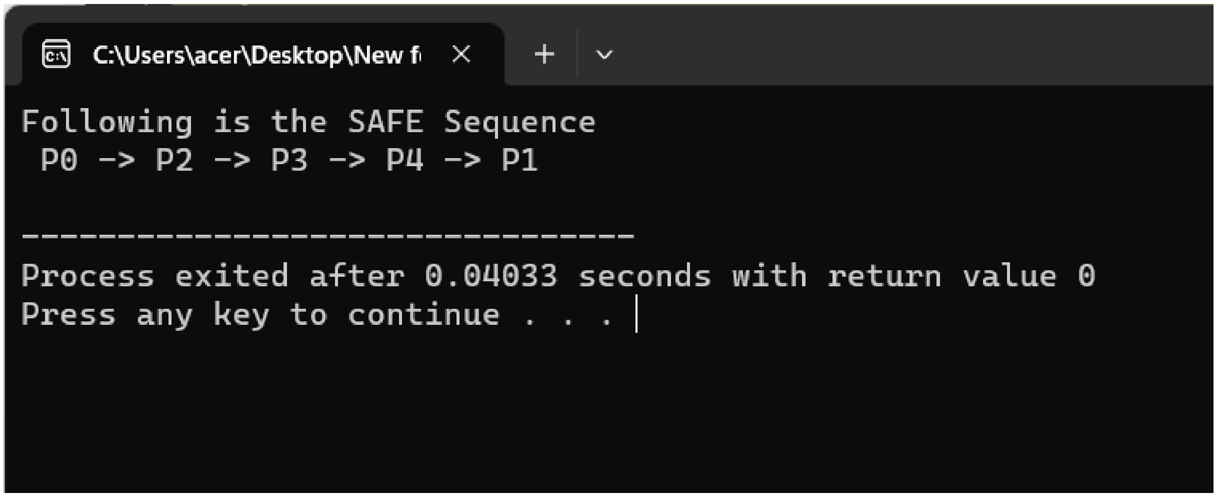
LAB 5 [SIMULATION OF THE BANKER’S ALGORITHM]

PROGRAM:

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

int main()
{
    int n, m, i, j, k;
    n = 5; // no. of processes
    m = 4; // no. of resources
    int alloc[5][4] = { { 0,0,1,2 }, // P0 // Allocation Matrix
                        { 1,0,0,0 }, // P1
                        { 1,3,5,4 }, // P2
                        { 0,6,3,2 }, // P3
                        { 0,0,1,4 } }; // P4
    int max[5][4] = { { 0,0,1,2 }, // P0 // MAX Matrix
                      { 1,7,5,0 }, // P1
                      { 2,3,5,6 }, // P2
                      { 0,6,5,2 }, // P3
                      { 0,6,5,6 } }; // P4
    int avail[4] = { 1,5,2,0 }; // Available Resources
    int f[n], ans[n], ind = 0;
    for (k = 0; k < n; k++)
    {
        f[k] = 0;
    }
    int need[n][m];
    for (i = 0; i < n; i++)
    {
        for (j = 0; j < m; j++)
        need[i][j] = max[i][j] - alloc[i][j];
    }
    int y = 0;
    for (k = 0; k < 5; k++)
    {
        for (i = 0; i < n; i++)
        {
            if (f[i] == 0)
            {
                int flag = 0;
                for (j = 0; j < m; j++)
                {
                    if (need[i][j] > avail[j])
                    {
                        flag = 1;
                        break;
                    }
                }
```

```
                        if (flag == 0)
                        {
                                ans[ind++] = i;
                                for (y = 0; y < m; y++)
                                        avail[y] += alloc[i][y];
                                f[i] = 1;
                        }
                }
        }
    }
    int flag = 1;

    // To check if sequence is safe or not
    for(int i = 0;i<n;i++)
    {
        if(f[i]==0)
        {
            flag = 0;
            cout << "The given sequence is not safe";
            break;
        }
    }

    if(flag==1)
    {
        cout << "Following is the SAFE Sequence" << endl;
        for (i = 0; i < n - 1; i++)
        cout << " P" << ans[i] << " ->";
        cout << " P" << ans[n - 1] <<endl;
    }
    return (0);
}
```

OUTPUT:



```
Following is the SAFE Sequence
 P0 -> P2 -> P3 -> P4 -> P1

--------------------------------
Process exited after 0.04033 seconds with return value 0
Press any key to continue . . .
```