**ASSIGNMENT 3**

**A3\_1] Implement the First come first serve scheduling algorithm using any programing language.**

#include<bits/stdc++.h>

#define ll long long int

#define fo(i,a,b) for(i=a;i<b;i++)

#define foe(i,a,b) for(i=a;i<=b;i++)

using namespace std;

int main()

{

ll n;

cout << "Enter number of processes : ";

cin >> n;

ll i,arr[n+1],burst[n+1],wait[n+1];

cout << "Enter burst times of n processes in order : ";

foe(i,1,n)

cin >> burst[i];

cout << "Enter order of processes : ";

foe(i,1,n)

cin >> arr[i];

wait[1] = 0;

float sum = 0;

foe(i,1,n-1)

{

wait[i+1] = wait[i] + burst[arr[i]];

sum += wait[i+1];

}

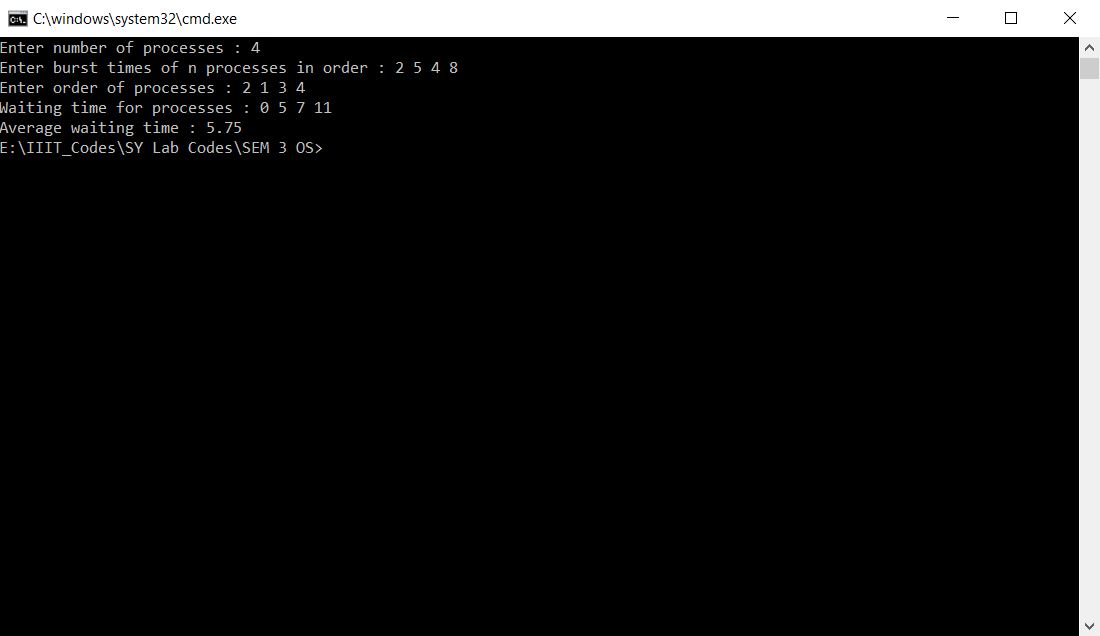
cout << "Waiting time for processes : ";

foe(i,1,n)

cout << wait[i] << ' ';

cout << "\nAverage waiting time : " << (float)(sum/n);

}



**A3\_2] Implement the Shortest job first scheduling algorithm using any programing language.**

#include<bits/stdc++.h>

#define ll long long int

#define fo(i,a,b) for(i=a;i<b;i++)

#define foe(i,a,b) for(i=a;i<=b;i++)

using namespace std;

int main()

{

ll n;

cout << "Enter number of processes : ";

cin >> n;

ll i,burst[n],wait[n];

cout << "Enter burst times of n processes in order : ";

fo(i,0,n)

cin >> burst[i];

sort(burst,burst+n);

wait[0] = 0;

float sum=0;

fo(i,1,n)

{

wait[i] = wait[i-1] + burst[i-1];

sum += wait[i];

}

cout << "Waiting time for processes : ";

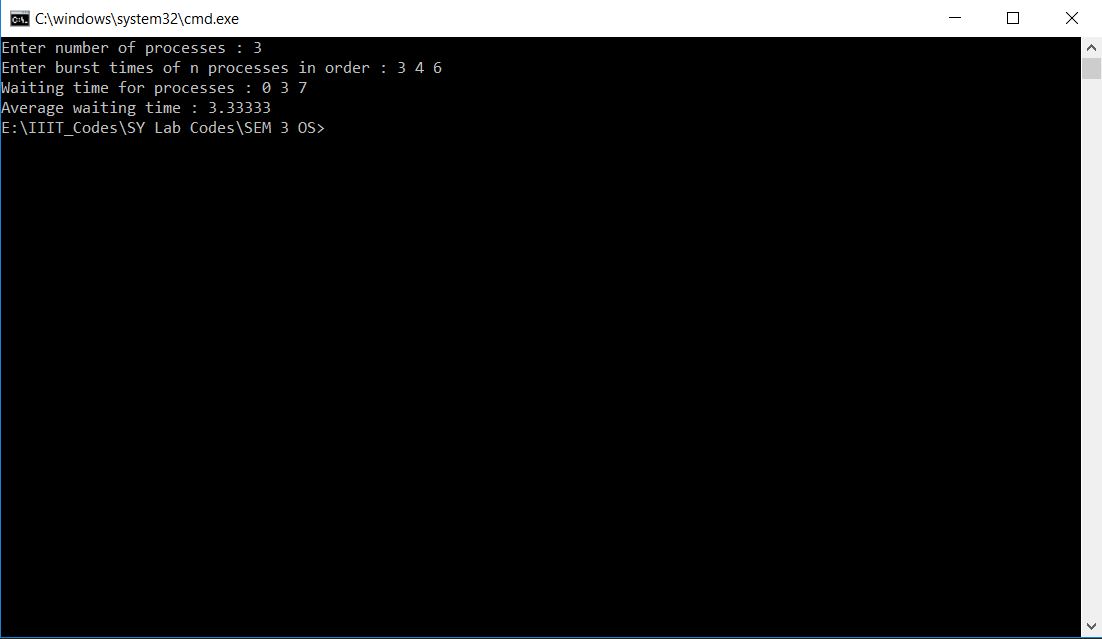
fo(i,0,n)

cout << wait[i] << ' ';

cout << "\nAverage waiting time : " << (float)(sum/n);

return 0;

}



**A3\_3] Implement the Round Robin scheduling algorithm using any programing language.**

#include<bits/stdc++.h>

#define ll long long int

#define mk make\_pair

#define pb push\_back

#define f first

#define s second

#define fo(i,a,b) for(i=a;i<b;i++)

#define foe(i,a,b) for(i=a;i<=b;i++)

using namespace std;

int main()

{

ll q,n,i,cnt=0, time = 0;

float sum=0;

cout << "Enter number of processes : ";

cin >> n;

vector <pair<ll,ll>> pro(n+1);

cout << "Enter pairs of (arrival times and Burst times) of n processes in order : \n";

foe(i,1,n)

{

ll a,b;

cin >> a >> b;

pro[i].f = a;

pro[i].s = b;

sum -= a;

sum -= b;

}

cout << "Enter time Quanta : ";

cin >> q;

vector <pair < ll, pair <ll,ll> > > wait;

wait.pb(mk(-1,mk(-1,-1)));

ll curr = 1,temp;

temp = curr;

ll pivot = -1;

while(cnt<n)

{

if((curr == pivot))

time++;

if(time < pro[temp].f)

{

pivot = temp;

temp = curr;

continue;

}

if(pro[temp].s == 0)

{

temp = (temp + 1 > n ? curr : temp + 1);

continue;

}

pivot = -1;

ll arrtime = time;

time += pro[temp].s > q ? q : pro[temp].s;

pro[temp].s -= pro[temp].s > q ? q : pro[temp].s;

wait.pb(mk(temp,mk(arrtime,time)));

if(pro[temp].s == 0)

{

cnt++;

sum += time;

}

temp = (temp + 1 > n ? curr : temp + 1);

if(pro[curr].s == 0)

curr += curr + 1 > n ? 0 : 1;

}

fo(i,1,wait.size())

{

cout << "P" << wait[i].f << ' ' << "Time : " << wait[i].s.f << " to " << wait[i].s.s << '\n';

}

cout << "Average Waiting time : " << (float)(sum/n);

return 0;

}

