**Experiment No:** 12

**Experiment name:**  Round Robin Scheduling Algorithm

**Advantages:**

i.This algorithm is very simple to implement.

ii.The aging technique is implemented to reduce the starvation.

**Source Code:**

#include<stdio.h>

int main()

{

int n,i,k,x=0,s=0,r=0,q=0,a[30],e[30],t[30];

float m,p=0;

printf("Enter the number of process: ");

scanf("%d",&n);

printf("Enter the execution time: ");

for(i=0; i<n; i++)

{

scanf("%d",&a[i]);

e[i]=a[i];

}

printf("Enter the quanta number: ");

scanf("%d",&q);

printf("After Round Robin sheduling : ");

for(i=0; i<n; i++)

{

if(x<a[i])

{

x=a[i];

}

}

k=x/q;

while(s<=k)

{

for(i=0; i<n; i++)

{

if(a[i]>0)

{

if(a[i]>q)

{

r=r+q;

a[i]=a[i]-q;

printf("P%d\t",i+1);

}else

{

r=r+a[i];

a[i]=a[i]-q;

printf("P%d ",i+1);

t[i]=r;

}

}

}

s++;

}

printf("\n\nProcess BurstTime WaitingTime TurnAroundTime\n");

for(i=0; i<n; i++)

{

printf(" %d \t\t %d\t\t %d\t\t %d\t\t \n",i,e[i],x,t[i]);

x=x+q;

}

m=x/n;

printf("\nAverage waiting time is =%f= ",m);

printf("\nAverage turn around time is = ");

for(i=0; i<n; i++)

p=p+t[i];

p=p/n;

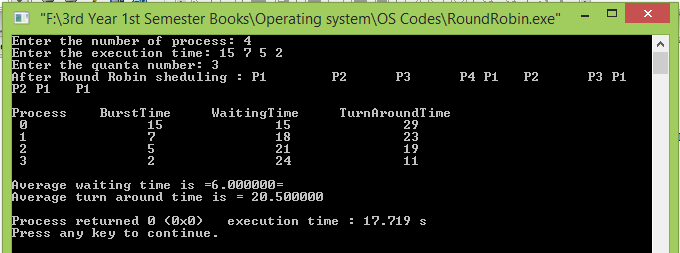
printf("%f",p);

printf("\n");

return 0;

}

**Result:**



**Conclusion:**

Round-robin scheduling results in max-min fairness if the data packets are equally sized.This type of scheduling is one of the very basic algorithm for operating systems in computer which can be implemented through circular queue data structure. Our honorable teacher helped us by giving some important lecture which helped us to code these above method.