Name: M.Nikhlesh

Reg No: 11602980

E-mail id: nikhlesh.rao@gmail.com

Git Hub link: https://github.com/Nikhlesh117/OS.git

Code:

```
#include<iostream>

#include<conio.h>

using namespace std;


void SearchStack01(int pnt,int tm);

void SearchStack02(int pnt, int tm);

void AddQue(int pnt);

int at[50], bt[50], ct[50]={0}, qt, rqi[50]={0}, c=0, st, flg=0, tm=0, noe=0, pnt=0, btm[50]={0}, tt, wt,priority[50];

float att, awt;


main(){


cout<<"ROUND ROBIN ALGO : INPUT 5 PROCESSES\n";

for(int x=0;x<5;x++){

   cout<<"\nProcess "<<x+1;

   cout<<"\nAT=";

   cin>>at[x];

   cout << "BT=";

   cin>>bt[x];

   btm[x]=bt[x];
```

```
cout<<"enter priority";
cin>>priority[x];
}

cout<<"\nEnter time quantum:";
cin>>qt;

system("CLS");

cout<<endl<<"GANTT CHART"<<endl<<at[0];
do{
   if(flg==0){
      st=at[0];

      if(btm[0]<=qt){
         tm=st+btm[0];
         btm[0]=0;
         SearchStack01(pnt,tm);}
      else{
         btm[0]=btm[0]-qt;
         tm=st+qt;
         SearchStack01(pnt,tm);
         AddQue(pnt);}
   }

   else{
      pnt=rqi[0]-1;
```

```
      st=tm;

      for(int x=0;x<noe && noe!=1;x++){
        rqi[x]=rqi[x+1];}
      noe--;

      if(btm[pnt]<=qt){
        tm=st+btm[pnt];
        btm[pnt]=0;
        SearchStack02(pnt, tm);}
      else{
       btm[pnt]=btm[pnt]-qt;
        tm=st+qt;
        SearchStack02(pnt, tm);
        AddQue(pnt);}
    }

       if(btm[pnt]==0){
      ct[pnt]=tm;
    }

   flg++;
   cout<<"]-P"<<pnt+1<<"-["<<tm;

}while(noe!=0);

cout<<"\n\nPROCESS\t AT\t BT\t CT\t TT\t WT\n";
```

```
for(int x=0;x<5;x++){
   tt=ct[x]-at[x];
   wt=tt-bt[x];
   cout<<"P"<<x+1<<" \t "<<at[x]<<" \t "<<bt[x]<<" \t "<<ct[x]<<" \t "<<tt<<" \t "<<wt<<"\n";
   awt=awt+wt;
   att=att+tt;
}

cout<<"\nAVERAGE TT: "<<att/5<<"\nAVERAGE WT: "<<awt/5;


}

void SearchStack01(int pnt,int tm){
   for(int x=pnt+1;x<5;x++){
      if(at[x]<=tm){
         rqi[noe]=x+1;
         noe++;}
   }
}

void SearchStack02(int pnt, int tm){
   for(int x=pnt+1;x<5;x++){

      int fl=0;
      for(int y=0;y<noe;y++){
         if(rqi[y]==x+1){
            fl++;}}
```

```
    if(at[x]<=tm && fl==0 && btm[x]!=0){
      rqi[noe]=x+1;
      noe++;}
  }
}

void AddQue(int pnt){
  rqi[noe]=pnt+1;
  noe++;
}
```

1. Explain the problem in terms of operating system concept?

Description:
Round-robin (RR) is one of the algorithms employed by process and network schedulers in computing. As the term is generally used, time slices (also known as time quanta)[3] are assigned to each process in equal portions and in circular order, handling all processes without priority (also known as cyclic executive). Round-robin scheduling is simple, easy to implement, and starvation-free. Round-robin scheduling can also be applied to other scheduling problems, such as data packet scheduling in computer networks. It is an operating system concept

Priority scheduling is a non-preemptive algorithm and one of the most common scheduling algorithms in batch systems. Each process is assigned a priority. Process with the highest priority is to be executed first and so on. Processes with the same priority are executed on first come first served basis. Priority can be decided based on memory requirements, time requirements or any other resource requirement.

The period of time for which a process is allowed to run in a preemptive multitasking system is generally called the *time slice* or *quantum*. The scheduler is run once every time slice to choose the next process to run. The length of each time slice can be critical to balancing system performance vs process responsiveness - if the time slice is too short then the scheduler will consume too much processing time, but if the time slice is too long, processes will take longer to respond to input.

2. Write the algorithm for proposed solution of the assigned problem.
   Algorithm:
   1. declare and initialize the array of burst time, arrival time, priority,and process.
   2. Sort the array of burst time, arrival time,priority and process on the basis of priority in the increasing order.
   3. apply Round Robin Algorithm.
   4. Calculate waiting time:
   Waiting time= Completion time - Burst time - arrival time.

3. Calculate complexity of implemented algorithm.

Description:
The overall complexity of code is n^2.

4. Explain all the test cases applied on the solution of assigned problem**.**

Description:
1.)
Time Quantum=2

Process	**Arrival Time**	**Burst Time**
P1	0	6
P2	1	4
P3	4	2

2.)
Time Quantum=5

Process	**Arrival Time**	**Burst Time**
P1	2	6
P2	1	10
P3	4	3

3.)
Time Quantum=4

Process	**Arrival Time**	**Burst Time**
P1	15	1
P2	10	2
P3	1	3

4.)
Time Quantum=5

Process	**Arrival Time**	**Burst Time**
P1	0	10
P2	1	15
P3	2	20

5.)
Time Quantum=10

Process	**Arrival Time**	**Burst Time**
P1	2	20
P2	9	40
P3	1	15

5. Explain the boundary conditions of the implemented code.

Description:

1.) Max process schedule at a time=50
2.) Time quantum should not be zero or negative.

6. Have you made minimum 5 revisions of solution on GitHub?

GitHub Link: https://github.com/Nikhlesh117/OS.git