## Student Name : Rishabh Sharma

## Student Id : 11713811

## Email Address : [rishabhsharma88.rs@gmail.com@gmail.com](mailto:rishabhsharma88.rs@gmail.com@gmail.com)

## Github Link :

## Code :

#include<stdio.h>

int cur\_time,burst=0;

struct process

{

int priority;

int arr\_time;

int burst\_time;

int pid;

int w\_time;

};

int t=-1,a[30],i=-1;

struct process q[30],result[30],swap;

int number()

{

int l;

printf("Enter the number of process you want to enter: ");

scanf("%d",&l);

return l;

}

void calc()

{

if(t>=0)

{

int w,e,r,y;

if(burst!=0 && q[0].burst\_time!=0)

{

q[0].burst\_time--;

burst--;

q[0].priority++;

q[0].arr\_time=cur\_time+1;

for(w=1;w<=t;w++)

{

q[w].priority+=2;

q[w].w\_time=++q[w].w\_time;

}

}

if(q[0].burst\_time==0)

{

i++;

result[i]=q[0];

for(w=0;w<t;w++)

{

q[w]=q[w+1];

}

t--;

}

for(w=0;w<t;w++)

{

for(e=0;e<t;e++)

{

if(q[w].priority<=q[e].priority)

{

swap=q[w];

q[w]=q[e];

q[e]=swap;

}

}

}

if(q[0].priority<=q[1].priority && t>=1)

{

swap=q[0];

for(w=0;w<t;w++)

{

q[w]=q[w+1];

}

q[t]=swap;

}

}

}

void main()

{

int l,min,n=number(),j,c=0;

float avg\_w\_time=0;

struct process p[n];

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

{

p[l].pid=l+1;

printf("\nEnter the arrival time of %d process: ",l+1);

scanf("%d",&p[l].arr\_time);

printf("\nEnter the burst time of %d process: ",l+1);

scanf("%d",&p[l].burst\_time);

p[l].priority=0;

p[l].w\_time=0;

burst=burst+p[l].burst\_time;

}

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

{

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

{

if(p[l].arr\_time<p[j].arr\_time)

{

swap=p[l];

p[l]=p[j];

p[j]=swap;

}

if(p[l].arr\_time==p[j].arr\_time)

{

if(p[l].burst\_time<p[j].burst\_time)

{

swap=p[l];

p[l]=p[j];

p[j]=swap;

}

}

}

}

printf("YOU ENTERED:\n\*(TABLE IS SORTED ACCORDING TO THE ARRIVAL TIME)\n\n");

printf(" PROCESS TABLE \n");

printf("----------------------------------------------\n");

printf(" PID ARRIVAL TIME SERVICE TIME \n");

printf("----------------------------------------------\n");

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

{

printf(" %d %d %d\n",p[l].pid,p[l].arr\_time,p[l].burst\_time);

}

cur\_time=p[0].arr\_time;

for(j=p[0].arr\_time;j<=p[n-1].arr\_time;j++)

{

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

{

if(p[l].arr\_time==j && c!=n)

{

t++;

q[t]=p[l];

c++;

}

if(c==n)

break;

}

calc();

cur\_time++;

while(burst!=0 && c==n)

{

calc();

cur\_time++;

}

if(c==n)

break;

}

printf("\nPROCESS IN ORDER OF THEIR COMPLETION:\n\n");

printf(" FINAL PROCESS EXECUTION TABLE \n");

printf("-------------------------------------------------------------------------------------\n");

printf(" PID ARRIVAL TIME COMPLETION TIME SERVICE TIME WAITING TIME \n");

printf("-------------------------------------------------------------------------------------\n");

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

{

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

{

if(result[l].pid==p[j].pid)

{

printf(" %d %d %d %d %d\n",result[l].pid,p[j].arr\_time,result[l].arr\_time,p[j].burst\_time,result[l].w\_time);

break;

}

}

avg\_w\_time+=(result[l].w\_time);

}

printf("AVERAGE WAITING TIME: %fs\n",avg\_w\_time/n);

}

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

Description:

The problem is to calculate the waiting time of the process and then calculate the average wait time of the process and the type of scheduling to be used is ‘preemptive priority scheduling’ and in this problem priority increases automatically for each process.For process waiting priority increases by 2 for each time quanta which is given as 1 and for the process executing priority increases by 1.and the process having highest number of priority has the highest priority.

1. Write the algorithm for proposed solution of the assigned problem?

Algorithm:

process p1[n];

queue(p1);

//operations on p1 suxch as decrement burst time.

Increment priority

Increment waiting time of the processes waiting in the queue.

Check for the burst time of the process if completed or not

If completed

Store in the result

Else sort the array of process according to the priority.

Then again repeat the steps till burst time of all the processes becomes zero.

Then from result get the information needed.

1. Calculate complexity of implemented algorithm.

Description: O(n^3).

1. Explain all the constraints given in the problem. Attach the code snippet of the implemented constraint.

Code snippet:

[Sorting according to arrival time.](#sortingarrival)

[Sorting according to priority.](#sortingpriority)

[Priority increment while executing.](#incrementexecute)

[Priority increment while waiting.](#incrementwaiting)

1. If you have implemented any additional algorithm to support the solution, explain the need and usage of the same.

Description: Sorting.

1. Explain the boundary conditions of the implemented code.

Description: Arrival time should be positive.

Burst time should be positive.

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

Description:

Process Arrival time Burst time

P1 0 4

P2 1 1

P3 2 2

P4 3 1

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

GitHub Link: