Student Name: Pawandeep Kaur

Student ID: 11804673

EmailAddress:
pawandeepk2222@gmail.com

Github:https://github.com/Pawandeepk819/operatingsystem.git

CODE:

```
#include <stdio.h>

#include <stdlib.h>

  #include <string.h>
  int main()
{
 FILE *fp = fopen("cpu_burst.txt", "r");
  int
bt[20],p[20],wt[20],tat[20],i=0,j,n=5,total=0,pos
,temp;
     float avg_wt,avg_tat;
  printf("\nReading CPU_BURST.txt File\n");
```

```
     //for(i=0;i<5;i++)
     while((getc(fp))!=EOF)
     {

         fscanf(fp, "%d", &bt[i]);
           if(bt[i]>0){
         p[i]=i+1;  i++;}          //contains process
number
}
n=i;
for(i=0;i<n;i++)

{
    pos=i;
    for(j=i+1;j<n;j++)
    {
        if(bt[j]<bt[pos])
            pos=j;
    }
    temp=bt[i];
    bt[i]=bt[pos];
    bt[pos]=temp;
  temp=p[i];
    p[i]=p[pos];
    p[pos]=temp;
}
wt[0]=0;  //waiting time for first process will be
zero
//calculate waiting time
for(i=1;i<n;i++)
{
    wt[i]=0;
    for(j=0;j<i;j++)
        wt[i]+=bt[j];
    total+=wt[i];
}
avg_wt=(float)total/n;      //average waiting time
```

```
total=0;
printf("\nProcess\t    Burst Time    \tWaiting
Time\tTurnaround Time");
   for(i=0;i<n;i++)
   {
      tat[i]=bt[i]+wt[i];     //calculate turnaround
time
      total+=tat[i];
      printf("\np%d\t\t  %d\t\t
%d\t\t\t%d",p[i],bt[i],wt[i],tat[i]);
   }
   avg_tat=(float)total/n;     //average turnaround
time
   printf("\n\nAverage Waiting
Time=%f",avg_wt);
   printf("\nAverage Turnaround
Time=%f\n",avg_tat);
   fclose(fp);
   return 0;
```

}

## 1. Explain the problem in terms of operating system concept? (Max 200 word)

In simple terms, turnaround time is the total time needed for an application to provide the required output to the user. From a batch system perspective, turnaround time can be considered the time taken in batch formation and printing of results. The concept of turnaround time overlaps with lead time and contrasts with the concept of cycle time. Turnaround time is expressed in terms of units of time for a specific system state and at times for a given algorithm.

Turnaround time varies for different applications and different programming languages.

Many factors influence turnaround time, such as:

- Memory needed for the application
- Execution time needed for the application
- Resources needed for the application
- Operating environment

Turnaround time is an important component in the design of microprocessors, especially for multiprocessor systems. Faster turnaround designs are preferred by hardware design companies, as they lead to faster performance and computing speeds.

**3. Calculate complexity of implemented algorithm. (Student must specify complexity of**

**each line of code along with overall complexity) .**

*As we can see in the program that there is 2 "for loops" the complexity of first loop is "O(n)" and the  complexity of the second "for loop" is O(sqrt(n)) because second for loop itrate upto "i/2". So the over all complexity of the above code is O(n(sqrt(n))).*

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

*As we know that it is the basic program then the constraints is* ***1<=n<=100000****.*

**5. Explain the boundary conditions of the implemented code.**

*The program will clear all the boundary condition except for "0" and "1" because the for loop variable starts from "2" and if we enter the value "0" and "1" then the loop will break and the variable come out of the loop.*

Student Name: Pawandeep Kaur

Student ID: 11804673

Email Address: pawandeepk2222@gmail.com

Github:https://github.com/Pawandeepk819/operatingsystem.git

Code:

|  |  |
|---|---|
| #include<stdio.h> |  |
|  | int n; |
|  | struct process |
|  | { |
|  |  |
|  | int p_no; |
|  |  |
|  | int arrival_t,burst_t,ct,wait_t,taround_time,p; |
|  |  |
|  | int flag; |
|  | }p_list[100]; |
|  | void Sorting() |
|  | { |

| | |
|---|---|
| | struct process p; |
| | |
| | int i, j; |
| | |
| | for(i=0;i<n-1;i++) |
| | |
| | { |
| | |
| | for(j=i+1;j<n;j++) |
| | |
| | { |
| | |
| | if(p_list[i].arrival_t > p_list[j].arrival_t) |
| | |
| | { |
| | |
| | p = p_list[i]; |

| | |
|---|---|
| | |
| | p_list[i] = p_list[j]; |
| | |
| | p_list[j] = p; |
| | |
| | } |
| | |
| | } |
| | |
| | } |
| | } |
| | int main() |
| | { |
| | |
| | int i,t=0,b_t=0,peak; |
| | |
| | int a[10]; |
| | |

| | |
|---|---|
| | float wait_time = 0, taround_time = 0, avg_w_t=0, avg_taround_time=0; |
| | |
| | printf("enter the no. of processes: "); |
| | |
| | scanf("%d",&n); |
| | |
| | for(i = 0; i < n; i++) |
| | |
| | { |
| | |
| | p_list[i].p_no = i+1; |
| | |
| | printf("\nEnter Details For P%d process:-\n", p_list[i].p_no); |
| | printf("Enter Arrival Time: "); |
| | scanf("%d", &p_list[i].arrival_t ); |
| | printf("Enter Burst Time: "); |

| | |
|---|---|
| | scanf("%d", &p_list[i].burst_t); |
| | p_list[i].flag = 0; |
| | b_t = b_t + p_list[i].burst_t; |
| | } |
| | Sorting(); |
| | for(int i=0;i<n;i++) |
| | { |
| | a[i]=p_list[i].burst_t; |
| | } |
| | p_list[9].burst_t = 9999; |
| | for(t = p_list[0].arrival_t; t <= b_t+1;) |
| | { |
| | peak = 9; |
| | for(i=0;i<n;i++) |
| | { |
| | if(p_list[i].arrival_t <= t && p_list[i].burst_t < p_list[peak].burst_t && p_list[i].flag != 1) |

| | |
|---|---|
| | { |
| | peak = i; |
| | } |
| | if(p_list[peak].burst_t==0 && p_list[i].flag != 1) |
| | { |
| | p_list[i].flag = 1; |
| | p_list[peak].ct=t;p_list[peak].burst_t= 9999; |
| | printf("P%d completes in %d\n",p_list[i].p_no,p_list[peak].ct); |
| | } |
| | } |
| | t++; |
| | (p_list[peak].burst_t)--; |
| | } |
| | for(i=0;i<n;i++) |
| | { |

| | |
|---|---|
| | p_list[i].taround_time=(p_list[i].ct)-(p_list[i].arrival_t); |
| | avg_taround_time=avg_taround_time+ p_list[i].taround_time; |
| | p_list[i].wait_t=((p_list[i].taround_time)-a[i]); |
| | avg_w_t=avg_w_t+p_list[i].wait_t; |
| | } |
| | printf("PNO\tAT\tCT\tTA\tWTt\n"); |
| | for(i=0;i<n;i++) |
| | { |
| | printf("P%d\t%d\t%d\t%d\t%d\n",p_list[i].p_no,p_list[i].arrival_t,p_list[i].ct, p_list[i].taround_time |
| | ,p_list[i].wait_t); |
| | } |
| | printf("Average Turn around Time: %f\t\n\n",avg_taround_time); |

| | |
|---|---|
| | printf("Average Waiting Time :\t %f\t\n",avg_w_t); |
| | } |

## 1. Explain the problem in terms of operating system concept? (Max 200 word)

In computing, turnaround time is the total time taken between the submission of a program/process/thread/task  for execution and the return of the complete output to the customer/user.It may vary for various programming languages depending on the developer of the software or the program. Turnaround time may simply deal with the total time it takes for a program to provide the required output to the user after the program is started.

Turnaround time is one of the  used to evaluate an operating system's scheduling algorithms.

In case of batch systems, turnaround time will include time taken in forming batches, batch execution and printing results.

With increasing computerization of analytical instruments the distinction between a computing context and a "non-computing" context is becoming semantic. An example of a "non-computing" context of turnaround time is the time a particular analysis in a laboratory, such as a medical laboratory, other commercial laboratories or a public health laboratory takes to result. Laboratories may publish an average turnaround time to inform their clients, e.g. a health care worker ordering the test, after what time a result can be expected. A prolonged turnaround time may give the requester a clue that a specimen was not received, that an analysis met with problems within the lab including that the result was unusual and the test was repeated for quality control.

**2. Calculate complexity of implemented algorithm. (Student must specify complexity of**

**each line of code along with overall complexity) .**

*In the above code there are total of "8 for loop" the complexity of each of them is*

*"1 for loop":-The time time complexity is "O(n)"*

*Basically the overall complexity of the program is :-***O(r*(P*P)).**

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

*As we know that it is the basic program then the constraints is **1<=n<=100000**.*

**4. Explain the boundary conditions of the implemented code.**

*When the process required resources is equal to max number of resource.*

*And the all the number should be positive integer.*