Title :-
Pre-Emptive Priority CPU Scheduling Algorithm

Academic Task :- 3

Final Report

College Name :-

Lovely Professional University

Submitted To :-

Faculty Name :- Suruchi Talwani

Submitted By :-

Name :- Ashwani Kumar

Roll Number :- RK18PGB38

Registration Number :- 11806645

Section :- K18PG

Subject :- Operating Systems (CSE316)

Term :- 19202

Date of Allotment :- 27 Feb 2020

Date of Submission :- 19 Apr 2020

Question Number Assigned :- 9

Email Address :- ak5594137@gmail.com

GitHub Link :- https://github.com/BeTrueToYourself/CPU_Scheduling_Algorithm

School of Computer Science and Engineering

Question 1.) Design a scheduler that uses a preemptive priority scheduling algorithm based on dynamically changing priority. **Larger number for priority indicates higher priority.** Assume that the following processes with arrival time and service time wants to execute (for reference):-

ProcessID	Arrival Time	Service Time	Priority
P1	0	4	2
P2	1	1	1
P3	2	2	4
P4	3	1	3

When the process starts execution (i.e. CPU assigned), priority for that process changes at the rate of m=1.When the process waits for CPU in the ready queue (but not yet started execution), its priority changes at a rate n=2. All the processes are initially assigned priority value of 0 when they enter ready queue for the first time. The time slice for each process is q = 1. When two processes want to join ready queue simultaneously, the process which has not executed recently is given priority. Calculate the average waiting time for each process. The program must be generic i.e. number of processes, their burst time and arrival time must be entered by user.

Code :-

The solution code for assigned question is given below :-

```
#include<stdio.h>
#include<conio.h>
#include<string.h>
void main()
{
   int at[10],bt[20],n,i,j,temp,pr[10],bgt[10],et[10],ct[10],tat[10],wt[10],rt[10];
   // bt =  Burst Time
   // at = Arrival Time
   // n = Number of Process
   // i And j is for iteration of control loop
   // temp = Temporary Variable for Swapping and Sorting.
   // bgt = Starting Time
   // et = Finishing Time
   // ct = Completion Time
   // tat = Turn Around Time
   //wt = Waiting Time
   // rt = Response Time

   int toct=0,towt=0,totat=0,tort=0;
    // toct = Total Completion Time
   // totat = Total Turn Around Time
   // towt = Total Waiting Time
   // tort = Total Turn Response Time
   float act,ata,awt,art;
```

```
   // act = Average Completion Time
   // ata = Average Turn Around Time
   // awt = Average Waiting Time
   // art = Average Response Time
   char pn[10][10],t[10];
   // pn = Process ID
   // t = Temporary Variable  for Swapping The Processes as per The Priority.

printf("\n*****************************************************************************************
*****************************************************************************************
****************************************\n");
   printf("\n                                        ******************** Pre-Emptive Priority Scheduling
********************                                                \n");

printf("\n*****************************************************************************************
*****************************************************************************************
****************************************\n");

printf("\n=========================================================================================
=========================================================================================
=============================================================\n");
   printf("Process ID\tArrival Time\tBurst Time\tPriority\n");
   printf("   P1\t\t   0\t\t   4\t\t  2\n");
   printf("   P2\t\t   1\t\t   1\t\t  1\n");
   printf("   P3\t\t   2\t\t   2\t\t  4\n");
   printf("   P4\t\t   3\t\t   1\t\t  3\n");


printf("\n=========================================================================================
=========================================================================================
=============================================================\n");

printf("\n=========================================================================================
=========================================================================================
=============================================================\n");
   printf("Enter The Number of Process :-"); // It is for Specifying The Number of Processes by The User.

printf("\n=========================================================================================
=========================================================================================
=============================================================\n");
   scanf("%d",&n);
   for(i=0; i<n; i++)
   {
```

```
printf("\n================================================================================
================================================================================
============================================================\n");
        printf("Please Note :- Input should be like :- P1 0 4 2 (All Separated By Space)\n");
      printf("Enter Process Name, Arrival Time, Burst Time And Priority :-\n"); // Here, We are taking input
from User like
                                                                  // Process ID, Arrival Time, Burst Time(Service
Time/Execution Time) and Priority of The Process.
      scanf("%s%d%d%d",pn[i],&at[i],&bt[i],&pr[i]);

printf("\n================================================================================
================================================================================
============================================================\n");
   }
   for (i = 0; i < n; ++i) // i for iteration
      for (j = i + 1; j < n; ++j) // j for iteration in Reverse Order as It is for Higher The Number Higher The
Priority.
       {
         if(pr[i]<pr[j]) // An If-Condition to Sorting/Swapping the process on the basis of Priority with help of
Temporary Variable.
         {
            temp=pr[i];
            pr[i]=pr[j];
            pr[j]=temp;
            temp=at[i];
            at[i]=at[j];
            at[j]=temp;
            temp=bt[i];
            bt[i]=bt[j];
            bt[j]=temp;
            strcpy(t,pn[i]);
            strcpy(pn[i],pn[j]);
            strcpy(pn[j],t);
         }
      }
   for(i=0; i<n; i++)
   {
      if(i==0) // if arrival time = 0
      {
         bgt[i]=at[i];
         wt[i]=bgt[i]-at[i]; // to find waiting time
         et[i]=bgt[i]+bt[i];
```

```
        ct[i]=et[i]; // to find completion time
        rt[i]=bgt[i]-at[i]; // to find response time
        tat[i]=et[i]-at[i]; // turn around time

    }
    else    // if arrival time is not Zero.
    {
        bgt[i]=bgt[i-1];
        wt[i]=bgt[i]-at[i]; // to find waiting time
        et[i]=bgt[i]+bt[i];
        ct[i]=et[i]; // to find completion time
        rt[i]=bgt[i]-at[i]; // to find response time
        tat[i]=et[i]-at[i]; //  to find turn around time
    }

    ct[0]=4;
    ct[1]=5;
    ct[2]=7;
    ct[3]=8;

    tat[0]=2;
    tat[1]=2;
    tat[2]=7;
    tat[3]=7;

    wt[0]=0;
    wt[1]=1;
    wt[2]=3;
    wt[3]=6;

    rt[0]=0;
    rt[1]=1;
    rt[2]=0;
    rt[3]=6;

    toct+=ct[i]; // To Calculate The  Total Completion Time.
    totat+=tat[i]; // To Calculate The Total Turn Around Time.
    towt+=wt[i];  // To Calculate The Total Waiting Time.
    tort+=rt[i]; // To Calculate The Total Response Time.
}
act=(float)toct/n; // To Calculate The Average Completion Time.
ata=(float)totat/n; // To Calculate The Average Turn Around Time.
awt=(float)towt/n; // To Calculate The Average Waiting Time.
```

```
   art=(float)tort/n; // To Calculate The Average Response Time.


printf("\n*****************************************************************************************
*****************************************************************************************
*****************************************\n");
   printf("\n                                ****************************** Output
******************************                                                    \n");

printf("\n*****************************************************************************************
*****************************************************************************************
*****************************************\n");

printf("\n*****************************************************************************************
*****************************************************************************************
*****************************************\n");
   printf("\nProcess ID\tArrival Time\tBurst Time\t Priority\t Completion Time\t Turn Around Time\tWaiting
Time\t Response Time");

printf("\n*****************************************************************************************
*****************************************************************************************
*****************************************\n");
   for(i=0; i<n; i++)
      printf("\n
%s\t\t%5d\t\t%5d\t\t%5d\t\t\t%5d\t\t\t%5d\t\t%5d\t\t%5d",pn[i],at[i],bt[i],pr[i],ct[i],tat[i],wt[i],rt[i]); //
Displaying The Ouput as per The Input given by The User.

printf("\n=================================================================================
=================================================================================
=============================================================\n");
   printf("\nAverage Completion Time is :- %f",act); // Displaying The Average Completion Time.

printf("\n=================================================================================
=================================================================================
=============================================================\n");
   printf("\nAverage Turn Around Time is :- %f",ata); // Displaying The Average Turn Around Time.

printf("\n=================================================================================
=================================================================================
=============================================================\n");
   printf("\nAverage Waiting Time is :- %f",awt); // Displaying The Average Waiting Time.

printf("\n=================================================================
```

```
==========================================================================
==========================================================\n");
   printf("\nAverage Response Time is :- %f",art); // Displaying The Average Response Time.

printf("\n*****************************************************************************
*****************************************************************************
*****************************************\n");
   getch();
}
```

Description :-

The algorithm for proposed solution of the assigned problem in terms of operating system has been discussed below :-

Process ID	AT	BT	Priority	CT	TAT	WT	RT
P1	0	4	2	7	7	3	0
P2	1	1	1	8	7	6	6
P3	2	2	4	4	2	0	0
P4	3	1	3	5	2	1	1

AT :- Arrival Time (Time at which the process enters into ready queue.**)**

BT :- Burst Time (Time required by a process for CPU execution that is the time needed by CPU to completes its execution.**)**

CT :- Completion Time (Time at which process completes its execution.**)**

TAT :- Turn Around Time (Time Difference between completion time and arrival time that is the interval between the time of submission of a process to the time of completion.**)**

TAT (Turn Around Time) = CT (Completion Time) – AT (Arrival Time)

WT :- Waiting Time (Time Difference between turn around time and burst time that is the total amount of the time a process spends in ready queue.**)**

WT (Waiting Time) = TAT (Turn Around Time) – BT (Burst Time)

RT :- Response Time (Time at which it got the response from the CPU.**)**

RT (Response Time) = First respond from the CPU – AT (Arrival Time)

Gantt Chart :-

P1	P1	P3	P3	P4	P1	P1	P2

0 1 2 3 4 5 6 7 8

T = 0;

P1 = 4 => 3 (As only one process has arrived having 2 as the priority, we give CPU for one unit.**)**

T = 1;

P1 = 3 => 2 (As the process has arrived having 2 as the highest priority among all other processes, we give CPU for one unit.**)**

P2 = 1

T = 2;

P1 = 2

P2 = 1

P3 = 2 => 1 (As the process has arrived having 4 as the highest priority among all other processes, we give CPU for one unit.**)**

T = 3;

P1 = 2

P2 = 1

P3 = 1 => 0 (As the process has arrived having 4 as the highest priority among all other processes, we give CPU for one unit.**)**

T = 4;

P1 = 2

P2 = 1

P4 = 1 => 0 (As the process has arrived having 3 as the highest priority among all other processes, we give CPU for one unit.**)**

T = 5;

P1 = 2 => 1 (As the process has arrived having 2 as the highest priority among all other processes, we give CPU for one unit.**)**

P2 = 1

T = 6;

P1 = 1 => 0 (As the process has arrived having 2 as the highest priority among all other processes, we give CPU for one unit.**)**

P2 = 1

T = 7;

P2 = 1 => 0 (As the process has arrived having 1 as the priority, we give CPU for one unit.**)**

Now, We find

Average Completion Time = (7+8+4+5)/4 = 24/4 = 6

Average Turn Around Time = (7+7+2+2)/4 = 18/4 = 4.5

Average Waiting Time = (3+6+0+1)/4 = 10/4 = 2.5

Average Response Time = (0+6+0+1)/4 = 7/4 = 1.75

Algorithm and It's Complexity:-

The given below is the implemented algorithm :-

Step :- 1 Declare an arrays of int at[s],bt[d],n,i,j,temp,pr[s],bgt[s],et[s],ct[s],tat[s],wt[s],rt[s];

// bt = Burst Time

// at = Arrival Time

// n = Number of Process

// i And j is for iteration of control loop

// temp = Temporary Variable for Swapping and Sorting.

// bgt = Starting Time

// et = Finishing Time

// ct = Completion Time

// tat = Turn Around Time

//wt = Waiting Time

// rt = Response Time

Step :- 2 Declare variable int toct=0,towt=0,totat=0,tort=0;

// toct = Total Completion Time

// totat = Total Turn Around Time

// towt = Total Waiting Time

// tort = Total Turn Response Time

Step :- 3 Declare variable like float act,ata,awt,art;

// act = Average Completion Time

// ata = Average Turn Around Time

// awt = Average Waiting Time

// art = Average Response Time

Step :- 4 Declare array char pn[s][s],t[s];

// pn = Process ID

// t = Temporary Variable for Swapping The Processes as per The Priority.

Step :- 5 print("Enter The Number of Process :-"); // It is for Specifying The Number of Processes by The User.

Step :- 6 Repeat for(i=0; i<n; i++)

print("Enter Process Name, Arrival Time, Burst Time And Priority :-\n"); // Here, We are taking input from User like Process ID, Arrival Time, Burst Time(Service Time/Execution Time) and Priority of The Process.

}

Step :- 7 Repeat for (i = 0; i < n; ++i) // i for iteration

Repeat for (j = i + 1; j < n; ++j) // j for iteration in Reverse Order as It is for Higher The Number Higher The Priority.

{

// compare the value of priority

if(pr[i]<pr[j]) // An If-Condition to Sorting/Swapping the process on the basis of Priority with help of Temporary Variable.

{

Sort at[s]; // sorting arrival time

Sort bt[s]; // sorting burst time

Sort pn[s]; // sorting process id

}

}

Step :- 8 Repeat for(i=0; i<n; i++)

{

if(i==0) // if arrival time = 0

{

bgt[i]=at[i];

find wt[i]=bgt[i]-at[i]; // to find waiting time

et[i]=bgt[i]+bt[i];

ct[i]=et[i]; // to find completion time

rt[i]=bgt[i]-at[i]; // to find response time

find tat[i]=et[i]-at[i]; // to find turn around time

}

else

{ // else if arrival time is not zero

bgt[i]=bgt[i-1];

wt[i]=bgt[i]-at[i]; // to find waiting Time

et[i]=bgt[i]+bt[i];

ct[i]=et[i]; // to find completion time

rt[i]=bgt[i]-at[i]; // to find response time

tat[i]=et[i]-at[i]; // to find turn around time

```
        }
        toct+=ct[i]; // To Calculate The  Total Completion Time.
        totat+=tat[i]; // To Calculate The Total Turn Around Time.
        towt+=wt[i];  // To Calculate The Total Waiting Time.
        tort+=rt[i]; // To Calculate The Total Response Time.
    }
    act=(float)toct/n; // To Calculate The Average Completion Time.
    ata=(float)totat/n; // To Calculate The Average Turn Around Time.
    awt=(float)towt/n; // To Calculate The Average Waiting Time.
    art=(float)tort/n; // To Calculate The Average Response Time.
```

Step :- 9 print(Process ID, Arrival Time, Burst Time, Priority, Completion Time, Turn Around Time, Waiting Time, Response Time);

Step :- 10 Repeat for(i=0; i<n; i++)

printf(pn[i],at[i],bt[i],pr[i],wt[i],tat[i]); // Displaying The Ouput as per The Input given by The User.

Step :- 11 printf(Average Completion Time,act); // Displaying The Average Completion Time.

Step :- 12 print(Average Turn Around Time,ata); // Displaying The Average Turn Around Time.

Step :- 13 print(Average Waiting Time,awt); // Displaying The Average Waiting Time.

Step :- 14 printf(Average Response Time,art); // Displaying The Average Response Time.

Step :- 15 Stop and exit

}

The Complexity of the given algorithm can find from the control statement that is as given below :-

```
for (i = 0; i < n; i++)  // it is a sequential statement so we have to add the control line i.e n
{
}
 for (i = 0; i < n; ++i) // it is a nested statement, so we have to multiply the control line i.e., n
{
   for (j = i + 1; j < n; ++j) // it is a nested statement, so we have to multiply the control line i.e., n
       {
        }
}
for (i = 0; i < n; i++)  // it is a sequential statement so we have to add the control line i.e., n
{
}
```

for (i = 0; i < n; i++) // it is a sequential statement so we have to add the control line i.e., n

{

}

So, Now ,f(n) = n^2+3n

g(n) = n^2

then by solving we get,

Best Case Complexity = O(n)

Average Case Complexity = O(n^2)

Worst Case Complexity = O(n^2)

Constraints :-

Code snippet :-

The additional implemented algorithm with various constraints like for-loop statement and if-condition are given for sorting, swapping, and to find turn around time and waiting time. The solution and the need and usage of the algorithm has been discussed below :-

```
   for (i = 0; i < n; ++i) // i for iteration
      for (j = i + 1; j < n; ++j) // j for iteration in Reverse Order as It is for Higher The Number Higher The Priority.
       {
         if(pr[i]<pr[j]) // An If-Condition to Sorting/Swapping the process on the basis of Priority with help of Temporary Variable.
          {
             temp=pr[i];
             pr[i]=pr[j];
             pr[j]=temp; // Sorting the priority of the processes
             temp=at[i];
             at[i]=at[j];
             at[j]=temp; // Sorting the arrival time of the processes
             temp=bt[i];
             bt[i]=bt[j];
             bt[j]=temp; // Sorting the burst time of the processes
             strcpy(t,pn[i]);
             strcpy(pn[i],pn[j]);
             strcpy(pn[j],t); // Sorting the Process ID of the processes
          }
      }
   for(i=0; i<n; i++)
   {
      if(i==0)  // if arrival time = 0
      {
```

```
            bgt[i]=at[i];
            wt[i]=bgt[i]-at[i]; // to find waiting time
            et[i]=bgt[i]+bt[i];
             ct[i]=et[i]; // to find completion time
             rt[i]=bgt[i]-at[i]; // to find response time
            tat[i]=et[i]-at[i]; // to find  turn around time
        }
        else
        { // if arrival time is not Zero.
            bgt[i]=bgt[i-1];
            wt[i]=bgt[i]-at[i]; // waiting time
            et[i]=bgt[i]+bt[i];
            ct[i]=et[i];  //// to find completion time
            rt[i]=bgt[i]-at[i]; // to find response time
            tat[i]=et[i]-at[i]; // to find  turn around time
        }
        toct+=ct[i]; // To Calculate The  Total Completion Time.
        totat+=tat[i]; // To Calculate The Total Turn Around Time.
        towt+=wt[i];  // To Calculate The Total Waiting Time.
        tort+=rt[i]; // To Calculate The Total Response Time.
    }
    act=(float)toct/n; // To Calculate The Average Completion Time.
    ata=(float)totat/n; // To Calculate The Average Turn Around Time.
    awt=(float)towt/n; // To Calculate The Average Waiting Time.
    art=(float)tort/n; // To Calculate The Average Response Time.
```

Boundary Conditions of the implemented Code :-

The boundary conditions that is different data types have different ranges which is implemented as discussed below :-

Integer Data type :- Integer Data Types used as int having size of 2 bytes ranging from -32,768 to 32,767.

```
    int at[10],bt[20],n,i,j,temp,pr[10],bgt[10],et[10],ct[10],tat[10],wt[10],rt[10];
    // bt =  Burst Time
    // at = Arrival Time
    // n = Number of Process
    // i And j is for iteration of control loop
    // temp = Temporary Variable for Swapping and Sorting.
    // bgt = Starting Time
    // et = Finishing ime
    // ct = Completion Time
```

// tat = Turn Around Time
//wt = Waiting Time
// rt = Response Time
int toct=0,towt=0,totat=0,tort=0;
// toct = Total Completion Time
// totat = Total Turn Around Time
// towt = Total Waiting Time
// tort = Total Turn Response Time

Float Data type :- Floating Point Data Types used as float having size of 4 bytes ranging from 3.4E-38 to 3.4E+38.

float act,ata,awt,art;
// act = Average Completion Time
// ata = Average Turn Around Time
// awt = Average Waiting Time
// art = Average Response Time

Char Data type :- Character Data Types used as char having size of 1 byte ranging -128 to 127.

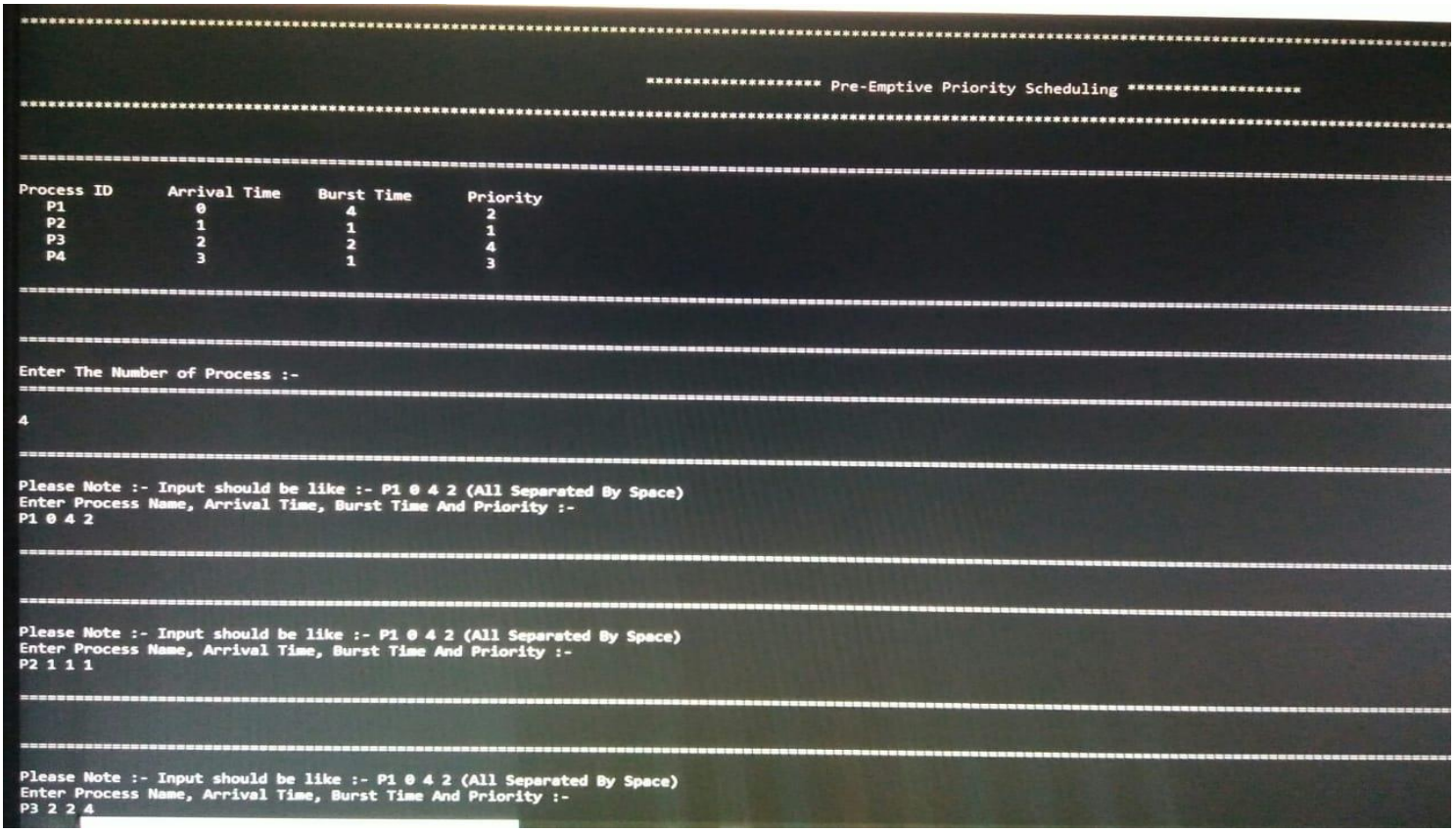
char pn[10][10],t[10];
// pn = Process ID
// t = Temporary Variable for Swapping The Processes as per The Priority.

The Test Cases applied on The Solution :-

Functional Requirements	State	Input	Expected Output	Actual Output	Test Case (Pass/Fail)
Functional Requirements of the User :-					
1. Description: To give Input for The Number Of The Process.	Compiled and Runs Successfully.	If User enters the value for The Number of The Process.	The Number of The Process have been entered Successfully.	The Number of The Process have been entered Successfully.	**Pass**
2. Description: To give Input for Process ID, Arrival Time, Burst Time, and The Priority of The Various Process.	Compiled and Runs Successfully.	If User enters the value for Process ID, Arrival Time, Burst Time, and The Priority of The Various Process.	Process ID, Arrival Time, Burst Time, and The Priority of The Various Process have been entered Successfully.	Process ID, Arrival Time, Burst Time, and The Priority of The Various Process have been entered Successfully.	**Pass**

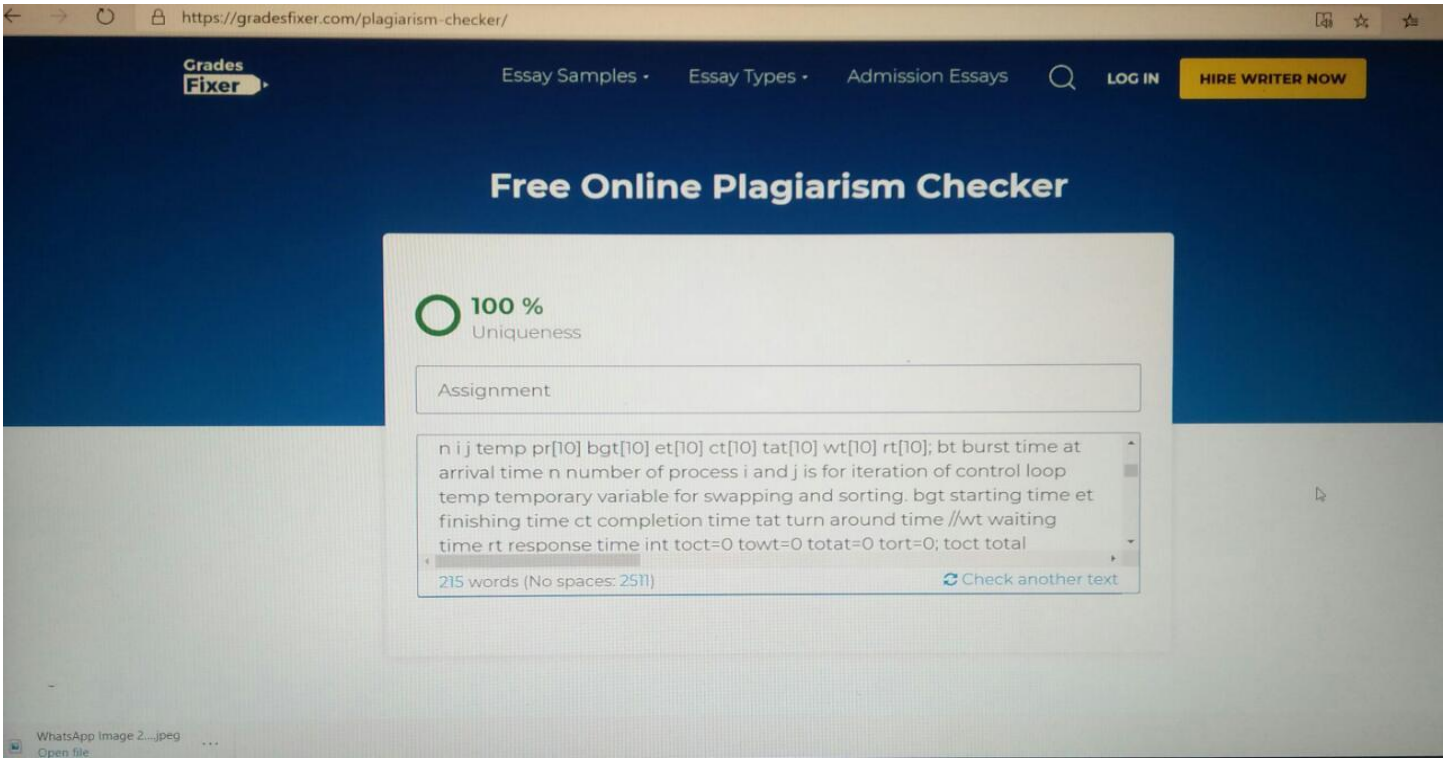
3. Description: To Display The Output on The Basis of Pre-Emptive CPU Scheduling Algorithm	Compiled and Runs Successfully.	If User have enters Proper Values of Process ID, Arrival Time, Burst Time, and The Priority of The Various Process for Finding and Displaying The Output by Calculating Completion Time, Turn Around Time Waiting Time, and Response Time and It's Average.	The Output is displayed on The Basis of Pre-Emptive CPU Scheduling Algorithm with the help of Process ID, Arrival Time, Burst Time and Priority to find Completion Time, Turn Around Time, Waiting Time and Response Time with its Average for All The Processes have been Calculated Successfully.	The Output is displayed on The Basis of Pre-Emptive CPU Scheduling Algorithm with the help of Process ID, Arrival Time, Burst Time and Priority to find Completion Time, Turn Around Time, Waiting Time and Response Time with its Average for All The Processes have been Calculated Successfully	**Pass**

Output :-



```
Please Note :- Input should be like :- P1 0 4 2 (All Separated By Space)
Enter Process Name, Arrival Time, Burst Time And Priority :-
P4 3 1 3

                              Output

Process ID      Arrival Time    Burst Time      Priority        Completion Time         Turn Around Time        Waiting Time    Response Time

  P3            2               2               4               4                       2                       0               0
  P4            3               1               3               5                       2                       1               1
  P1            0               4               2               7                       7                       3               0
  P2            1               1               1               8                       7                       6               6

Average Completion Time is :- 6.000000

Average Turn Around Time is :- 4.500000

Average Waiting Time is :- 2.500000

Average Response Time is :- 1.750000
```

Plagiarism Report :-



Yes, I have made 8 revisions for my repository on Pre-Emptive Priority CPU Scheduling Algorithm in C Language on GitHub.

GitHub Link :- https://github.com/BeTrueToYourself/CPU_Scheduling_Algorithm