Student Name: Pragy KumarVashistha

Reg. ID: 11803225 (B51)

Email Address: pragy3032@gmail.com

GitHub Link: https://github.com/fullmetal-101

Problem:

Ques1. Considering 4 processes with the arrival time and the burst time requirement of the processes the scheduler schedules the processes by interrupting the processor after every 3 units of time and does consider the completion of the process in this iteration. The schedulers then checks for the number of processes waiting for the processor and allots the processor to the process but interrupting the processor after every 6 units of time and considers the completion of the process in this iteration. The scheduler after the second iteration checks for the number of processes waiting for the processor and now provides the processor to the process with the least time requirement to go in the terminated state. The inputs for the number of requirements, arrival time and burst time should be provided by the user.
Consider the following units for reference.

Process	Arrival time	Burst time
P1	0	18
P2	2	23
P3	4	13
P4	13	10

Develop a scheduler which submits the processes to the processor in the defined scenario, and compute the scheduler performance by providing the waiting time for process, turnaround time for process and average waiting time and turnaround time.

Ans: We can solve this question by using algorithm of Operating System:

1. Round Robin Scheduling Algorithm : for fixed time slice i.e for X unit and Y unit oftime.

Algorithm:

```
time_req = 0;

// Add time for process on left of p

// (Scheduled before p in a round of

// 1 unit time slice)
```

```
                for (int i=0; i<p; i++)
                        {
                  if (arr[i] < arr[p])
                   time_req += arr[i];
                        else
                   time_req += arr[p];
                        }

          // step 2 : Add time of process p
                 time_req += arr[p];

           // Add time for process on right
             // of p (Scheduled after p in
           // a round of 1 unit time slice)
              for (int i=p+1; i<n; i++)
                        {
                  if (arr[i] < arr[p])
                   time_req += arr[i];
                        else
                  time_req += arr[p]-1;
                        }
```

.

Description:

To implement the above problem we have to make three iterations in which first iteration with a time slice of X units reduce the burst time of each process by X. In second iteration ------- --- --- ------ ----- ---- waiting time of each process.

Code:

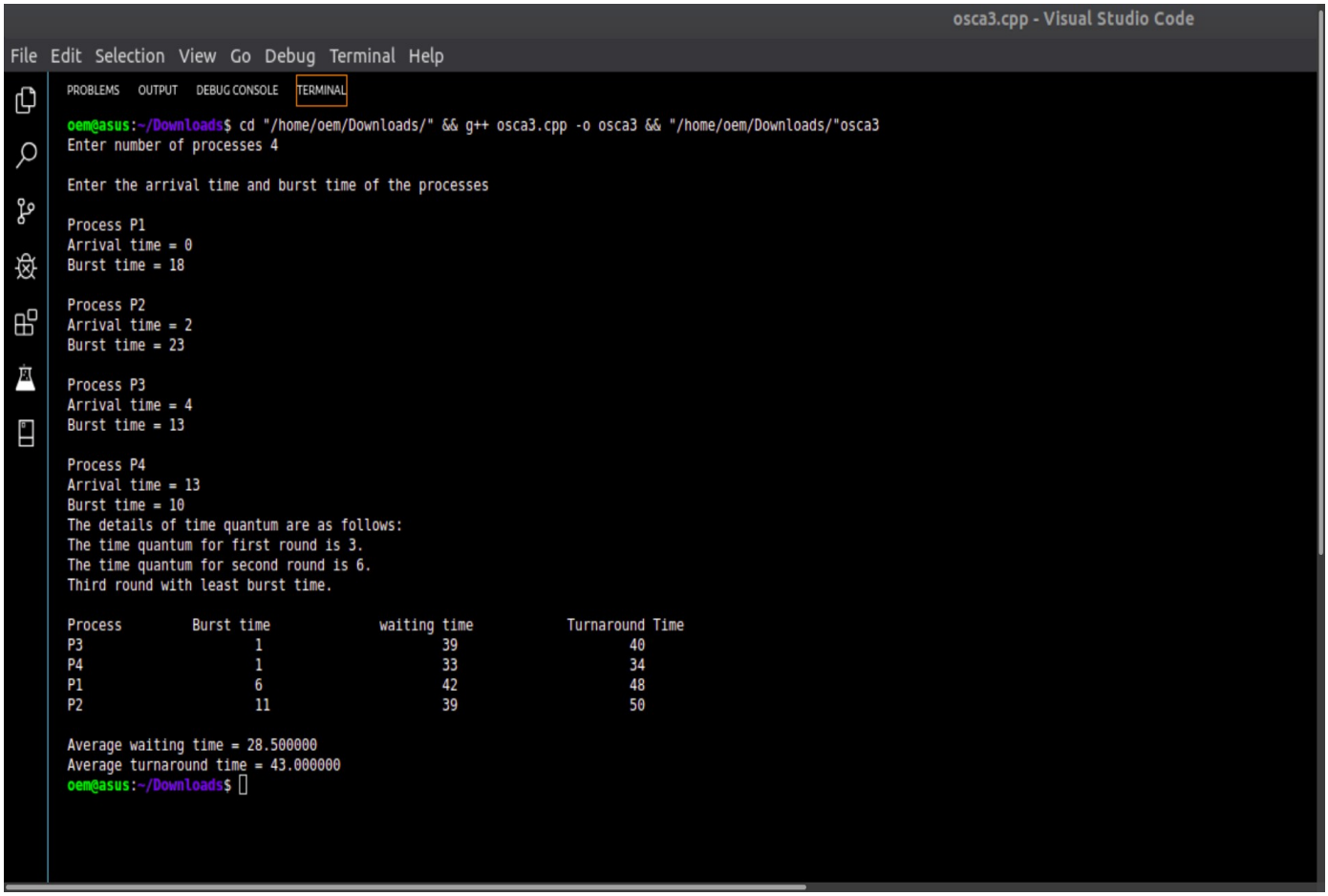
```
#include <stdio.h>

void rr(int no, int remt[10], int Cur_t, int arivalT[10], int burst
T[10]);

int main()
{
```

```
  int P_no, j, no, CurT, RemProc, indicator, time_Q, wait, tut, a
rivalT[10], burstT[10], remt[10], x = 1;
  indicator = 0;
  wait = 0;
  tut = 0;
  printf("Enter number of processes ");
  scanf("%d", &no);
  RemProc = no;
  printf("\nEnter the arrival time and burst time of the proces
ses\n");
  for (P_no = 0; P_no < no; P_no++)
  {
    printf("\nProcess P%d\n", P_no + 1);
    printf("Arrival time = ");
    scanf("%d", &arivalT[P_no]);
    printf("Burst time = ");
    scanf("%d", &burstT[P_no]);
    remt[P_no] = burstT[P_no];
  }
  printf("The details of time quantum are as follows:\n");
  printf("The time quantum for first round is 3.\n");
  time_Q = 3;
  CurT = 0;
  for (P_no = 0; RemProc != 0;)
  {
    if (remt[P_no] <= time_Q && remt[P_no] > 0)
    {
      CurT += remt[P_no];
      remt[P_no] = 0;
      indicator = 1;
    }
    else if (remt[P_no] > 0)
    {
      remt[P_no] -= time_Q;
      CurT += time_Q;
    }
```

```
    if (remt[P_no] == 0 && indicator == 1)
    {
        printf("%d", P_no);
        RemProc--;
        printf("P %d", P_no + 1);
        printf("\t\t\t%d", CurT - arivalT[P_no]);
        printf("\t\t\t%d\n", CurT - burstT[P_no] - arivalT[P_no]);
        wait += CurT - arivalT[P_no] - burstT[P_no];
        tut += CurT - arivalT[P_no];
        indicator = 0;
    }
    if (P_no == no - 1)
    {
        x++;
        if (x == 2)
        {
            P_no = 0;
            time_Q = 6;

            printf("The time quantum for second round is 6. \n");
        }
        else
        {
            break;
        }
    }
    else if (CurT >= arivalT[P_no + 1])
    {
        P_no++;
    }
    else
    {
        P_no = 0;
    }
}
```

```
  rr(no, remt, CurT, arivalT, burstT);

  return 0;
}

void rr(int no, int remt[10], int Cur_t, int arivalT[10], int burst
T[10])
{

  float avg_wait, avg_tut;
  int i, j, n = no, temp, btime[20], P_no[20], w_time[20], tut_t[
20], total = 0, loc;

  printf("Third round with least burst time.\n");

  for (i = 0; i < n; i++)
  {
    btime[i] = remt[i];
    w_time[i] = Cur_t - arivalT[i] - btime[i];
    P_no[i] = i + 1;
  }

  for (i = 0; i < n; i++)
  {
    loc = i;
    for (j = i + 1; j < n; j++)
    {
      if (btime[j] < btime[loc])
      {
        loc = j;
      }
    }
    temp = btime[i];
    btime[i] = btime[loc];
    btime[loc] = temp;
    temp = P_no[i];
```

```
    P_no[i] = P_no[loc];
    P_no[loc] = temp;
  }

  for (i = 1; i < n; i++)
  {
    for (j = 0; j < i; j++)
    {
      w_time[i] += btime[j];
    }
    total += w_time[i];
  }

  avg_wait = (float)total / n;
  total = 0;
  printf("\nProcess\t\tBurst time\t\twaiting time\t\tTurnaround
Time");
  for (i = 0; i < n; i++)
  {
    tut_t[i] = btime[i] + w_time[i];
    total = total + tut_t[i];
    printf("\nP%d\t\t\t%d\t\t\t%d\t\t\t%d", P_no[i], btime[i], w_t
ime[i], tut_t[i]);
  }
  avg_tut = (float)total / n;
  printf("\n\nAverage waiting time = %f", avg_wait);
  printf("\nAverage turnaround time = %f\n", avg_tut);
}
```


osca3.cpp - Visual Studio Code
File Edit Selection View Go Debug Terminal Help
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
oem@asus:~/Downloads$ cd "/home/oem/Downloads/" && g++ osca3.cpp -o osca3 && "/home/oem/Downloads/"osca3
Enter number of processes 4
Enter the arrival time and burst time of the processes
Process P1
Arrival time = 0
Burst time = 18
Process P2
Arrival time = 2
Burst time = 23
Process P3
Arrival time = 4
Burst time = 13
Process P4
Arrival time = 13
Burst time = 10
The details of time quantum are as follows:
The time quantum for first round is 3.
The time quantum for second round is 6.
Third round with least burst time.
Process Burst time waiting time Turnaround Time
P3 1 39 40
P4 1 33 34
P1 6 42 48
P2 11 39 50
Average waiting time = 28.500000
Average turnaround time = 43.000000
oem@asus:~/Downloads$