GROUP 1

SHAHZAIB HAMZA
(FA20-BSSE-0062)

SARFARAZ ILAHI SOOMRO
(FA20-BSSE-0027)

OS PROJECT PHASE 1

SUBMITTED TO MISS SUMRA KHAN

CODE:

OS Phase 1.cpp

```cpp
//os project p1
//shahzaib hamza
//sarfaraz soomro

#include <iostream> //standard input output stream
#include <algorithm>  //allow for sorting searching etc
#include <iomanip> //manupulating the output
#include <string.h>  //string usability
using namespace std;

struct process {
    int pno;  //process id
    int atime; //arrival time
    int btime; //burst time
    int priority; //priority
    int c_time; //completion time
    int t_time; //turn around time
    int w_time; //waiting time
    int r_time; //response time
    int start_time; //when process started
};
```

```cpp
//================================================================================

int main() {
printf("----OS Project Phase 1----(Shahzaib Hamza/Sarfaraz Soomro)\n\n");


    int num;
    int tit=0; //idle time of cpu when no process enter
    struct process p[100]; //max 100
    float avg_tatime; //avg turn around time
    float avg_wtime; //avg waiting time
    float avg_rtime; //avg respose time
    int total_ttime = 0; //total turn around time
    int total_wtime = 0; //total waiting time
    int total_rtime = 0; //total response time
    int burst_remaining[100]; //remaining burst time for premptive
    int is_completed[100]; //check process completion
    memset(is_completed,0,sizeof(is_completed)); //memset is used to fill a bl

//
```

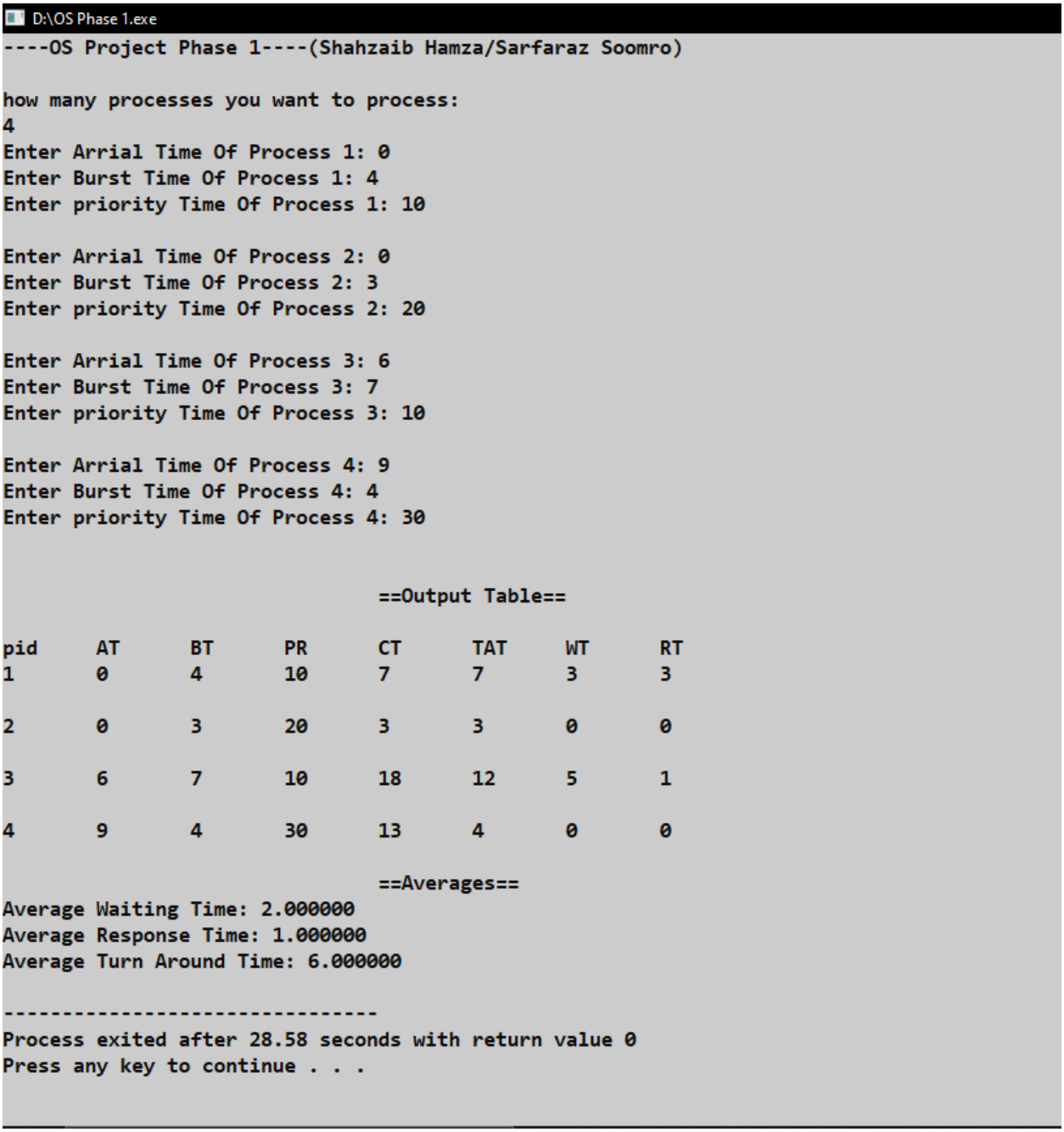
```
41  //=========================================================================
42  //input processes and details of processes
43
44   printf("how many processes you want to process: \n");
45      scanf("%d",&num);
46
47      for(int i = 0; i < num; i++) {
48
49          printf("Enter Arrial Time Of Process %d: ",i+1);
50          scanf("%d",&p[i].atime);
51
52          printf("Enter Burst Time Of Process %d: ",i+1);
53          scanf("%d",&p[i].btime);
54
55          printf("Enter priority Time Of Process %d: ",i+1);
56          scanf("%d",&p[i].priority);
57
58          p[i].pno=i+1;
59          burst_remaining[i] = p[i].btime;
60          printf("\n");
61
62      }
```

```
63  //=========================================================================
64  //main Logic
65
66   int cur_time = 0; //current time of proces
67      int cmpltd = 0; //process completed
68      int prev = 0; //previous process
69
70      while(cmpltd != num) {
71          int idc = -1; //current process id
72          int max = -1; //max priority process id
73
74          for(int i = 0; i < num; i++) {
75              if(p[i].atime <= cur_time && is_completed[i] == 0) {
76                  if(p[i].priority > max) {
77                      max = p[i].priority;
78                      idc = i;
79                  }
80                  if(p[i].priority == max) {
81                      if(p[i].atime < p[idc].atime) {
82                          max = p[i].priority;
83                          idc = i;
84                      }
85                  }
86              }
87          }
88
```

```
        if(idc != -1) {
            if(burst_remaining[idc] == p[idc].btime) {
                p[idc].start_time = cur_time;
                tit += p[idc].start_time - prev;
            }
            burst_remaining[idc] -= 1;
            cur_time++;
            prev = cur_time;

            if(burst_remaining[idc] == 0) {
                p[idc].c_time = cur_time;
                p[idc].t_time = p[idc].c_time - p[idc].atime;
                p[idc].w_time = p[idc].t_time - p[idc].btime;
                p[idc].r_time = p[idc].start_time - p[idc].atime;

                total_ttime += p[idc].t_time;
                total_wtime += p[idc].w_time;
                total_rtime += p[idc].r_time;

                is_completed[idc] = 1;
                cmpltd++;
            }
        }
        else {
             cur_time++;
        }
    }
```

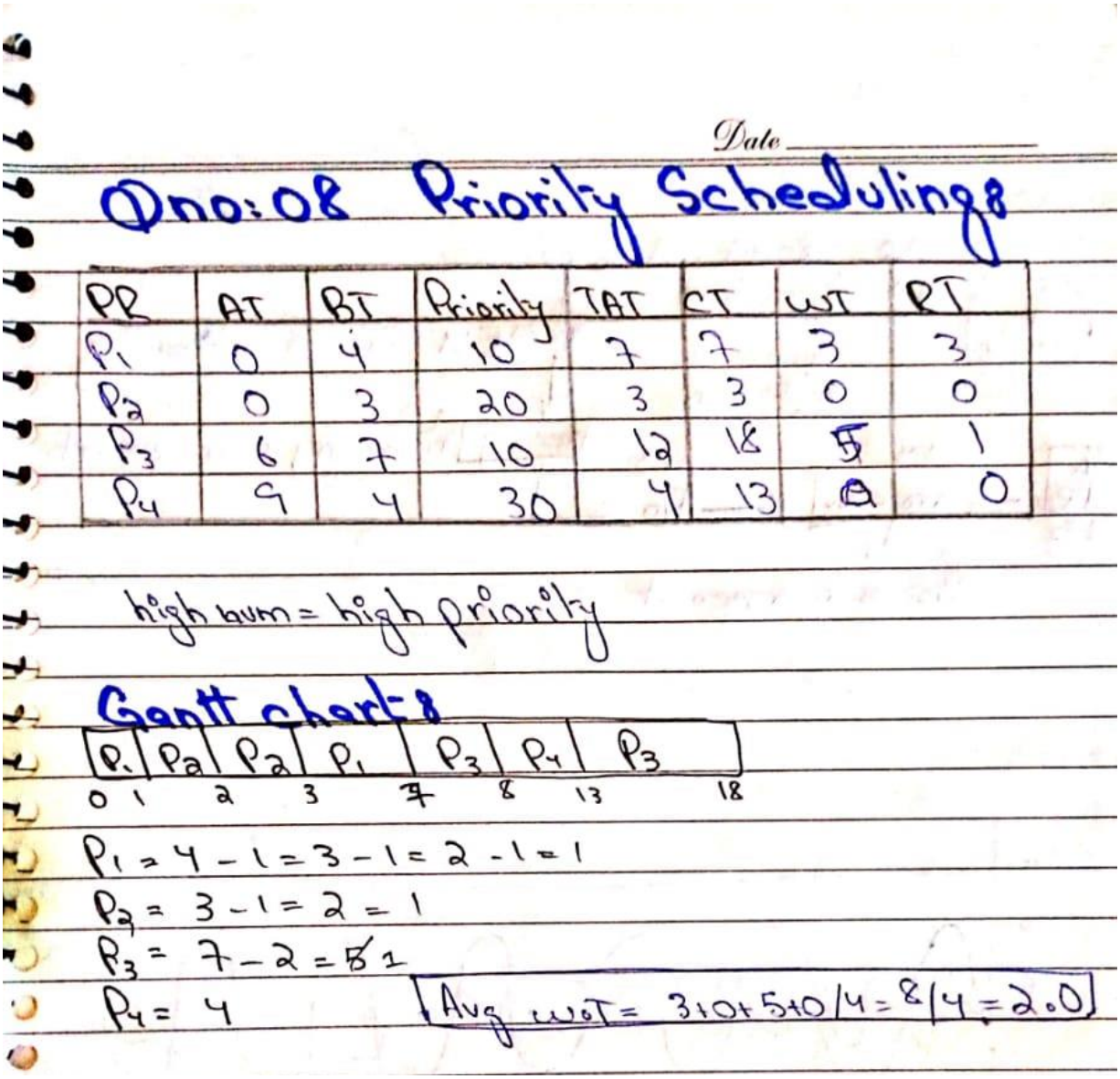
```
    //=================================================================================
    //formulas
    avg_tatime = total_ttime / num; //calculating average turn around time
    avg_wtime =(float) total_wtime / num;  //calculating average waiting timr
    avg_rtime = total_rtime / num;  //calculating average response time

    //=================================================================================
    //printing Output table
    printf("\n\t\t\t\t==Output Table==\n");
    printf("\npid\tAT\tBT\tPR\tCT\tTAT\tWT\tRT");
     for(int i = 0; i < num; i++) {
    printf("\n");
    printf("%d\t%d\t%d\t%d\t%d\t%d\t%d\t%d",p[i].pno,p[i].atime,p[i].btime,p[i].priority,p[i
    printf("\n");
    }

    //=================================================================================
    //Final Output
    printf("\n\t\t\t\t==Averages==\n");
    printf("Average Waiting Time: %f\n",avg_wtime);
    printf("Average Response Time: %f\n",avg_rtime);
    printf("Average Turn Around Time: %f\n",avg_tatime);
    }//main end
```

OTPUT



```
D:\OS Phase 1.exe
----OS Project Phase 1----(Shahzaib Hamza/Sarfaraz Soomro)

how many processes you want to process:
4
Enter Arrial Time Of Process 1: 0
Enter Burst Time Of Process 1: 4
Enter priority Time Of Process 1: 10

Enter Arrial Time Of Process 2: 0
Enter Burst Time Of Process 2: 3
Enter priority Time Of Process 2: 20

Enter Arrial Time Of Process 3: 6
Enter Burst Time Of Process 3: 7
Enter priority Time Of Process 3: 10

Enter Arrial Time Of Process 4: 9
Enter Burst Time Of Process 4: 4
Enter priority Time Of Process 4: 30


                            ==Output Table==

pid     AT      BT      PR      CT      TAT     WT      RT
1       0       4       10      7       7       3       3

2       0       3       20      3       3       0       0

3       6       7       10      18      12      5       1

4       9       4       30      13      4       0       0

                            ==Averages==
Average Waiting Time: 2.000000
Average Response Time: 1.000000
Average Turn Around Time: 6.000000

--------------------------------
Process exited after 28.58 seconds with return value 0
Press any key to continue . . .
```

REFERENCE ALGORITHM:



Date

Qno: 08 Priority Scheduling:

PR	AT	BT	Priority	TAT	CT	WT	RT
P_1	0	4	10	7	7	3	3
P_2	0	3	20	3	3	0	0
P_3	6	7	10	12	18	5	1
P_4	9	4	30	4	13	0	0

high num = high priority

Gantt chart:

P_1	P_2	P_2	P_1	P_3	P_4	P_3
0	1	2	3	7	8	13	18

$P_1 = 4 - 1 = 3 - 1 = 2 - 1 = 1$

$P_2 = 3 - 1 = 2 = 1$

$P_3 = 7 - 2 = 5 = 1$

$P_4 = 4$

Avg WT = 3+0+5+0/4 = 8/4 = 2.0