Name : Bijay Regmi
Registration Number : 210913032

Week 7 Lab Assignment

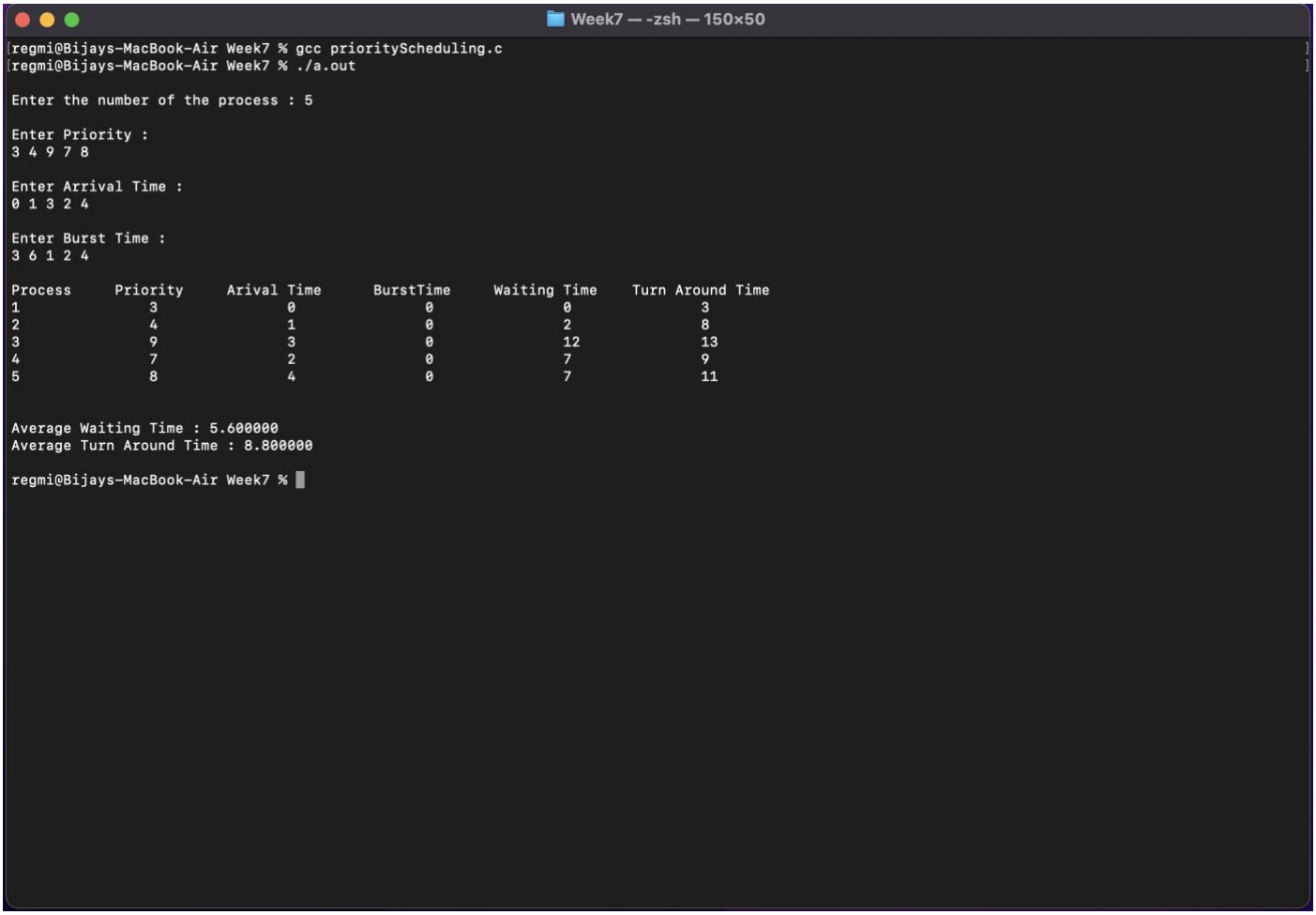
1. **WAP to illustrate priority scheduling in operation. The program accepts the total number of priorities, along with each packet priority, arrival time and burst time, and calculates the packet waiting time and turnaround time. Display all the timings and priority for each packet. Assume a non-preemptive priority queuing.**

```
#include <stdio.h>
struct process
{
   int WT, AT, BT, TAT, PT;
};

struct process a[10];

int main()
{
   int n, temp[10], t, count = 0, short_p;
   float total_WT = 0, total_TAT = 0, Avg_WT, Avg_TAT;
   printf("\nEnter the number of the process : ");
   scanf("%d", &n);
   printf("\nEnter Priority : \n");
   for (int i = 0; i < n; i++)
      scanf("%d",&a[i].PT);
   printf("\nEnter Arrival Time : \n");
   for (int i = 0; i < n; i++)
      scanf("%d",&a[i].AT);
   printf("\nEnter Burst Time : \n");
   for (int i = 0; i < n; i++){
      scanf("%d",&a[i].BT);
      // copying the burst time in
```

```
    // a temp array fot futher use
    temp[i] = a[i].BT;
}

// we initialize the burst time
// of a process with maximum
a[9].PT = 10000;

for (t = 0; count != n; t++)
{
    short_p = 9;
    for (int i = 0; i < n; i++)
    {
        if (a[short_p].PT > a[i].PT && a[i].AT <= t && a[i].BT > 0)
        {
            short_p = i;
        }
    }

    a[short_p].BT = a[short_p].BT - 1;

    // if any process is completed
    if (a[short_p].BT == 0)
    {
        // one process is completed
        // so count increases by 1
        count++;
        a[short_p].WT = t + 1 - a[short_p].AT - temp[short_p];
        a[short_p].TAT = t + 1 - a[short_p].AT;

        // total calculation
        total_WT = total_WT + a[short_p].WT;
        total_TAT = total_TAT + a[short_p].TAT;
    }
}

Avg_WT = total_WT / n;
Avg_TAT = total_TAT / n;
```

```
    // printing of the answer
    printf("\nProcess    Priority    Arival Time    BurstTime\tWaiting Time\tTurn Around Time");
    for (int i = 0; i < n; i++)
        printf("\n%d\t\t%d\t\t%d\t\t%d\t\t%d\t\t%d", i + 1, a[i].PT, a[i].AT, a[i].BT, a[i].WT, a[i].TAT);


    printf("\n\n\nAverage Waiting Time : %f", Avg_WT);
    printf("\nAverage Turn Around Time : %f\n\n", Avg_TAT);

    return 0;
}
```

INPUT/ OUTPUT



2. **WAP to illustrate round robin scheduling in operation. The program accepts the total number of classes, along with each packet class, arrival time and burst time. The program calculates the packet departure time, delay between arrival and departure time, and the average delay for all packets and displays all these timings along with the class for each packet. Assume a work conserving policy.**

```
#include <stdio.h>

int main()
{
   // initlialize the variable name
   int i, NOP, sum = 0, count = 0, y, quant, wt = 0, tat = 0, at[10], bt[10],
temp[10];
   float avg_wt, avg_tat;
   printf(" \nEnter total number of Processes : ");
   scanf("%d", &NOP);
   y = NOP; // Assign the number of process to variable y

   // Use for loop to enter the details of the process like Arrival time and the
Burst Time
   printf("\nEnter Arrival Time : \n");
   for (i = 0; i < NOP; i++)
      scanf("%d", &at[i]);
   printf("\nEnter Burst Time : \n");
   for (i = 0; i < NOP; i++){
      scanf("%d", &bt[i]);
      temp[i] = bt[i];
   }
   // Accept the Time qunat
   printf("\nEnter Time Quantum : ");
   scanf("%d", &quant);
   // Display the process No, burst time, Turn Around Time and the waiting time

   printf("\nProcess    Arrival Time    Burst Time    Turn Around Time     Waiting
Time ");
   for (sum = 0, i = 0; y != 0;)
```

```
{
   if (temp[i] <= quant && temp[i] > 0) // define the conditions
   {
      sum = sum + temp[i];
      temp[i] = 0;
      count = 1;
   }
   else if (temp[i] > 0)
   {
      temp[i] = temp[i] - quant;
      sum = sum + quant;
   }
   if (temp[i] == 0 && count == 1)
   {
      y--; // decrement the process no.
      printf("\n   %d          %d          %d          %d          %d",
       i + 1, at[i], bt[i], sum - at[i], sum - at[i] - bt[i]);
      wt = wt + sum - at[i] - bt[i];
      tat = tat + sum - at[i];
      count = 0;
   }
   if (i == NOP - 1)
   {
      i = 0;
   }
   else if (at[i + 1] <= sum)
   {
      i++;
   }
   else
   {
      i = 0;
   }
}
// represents the average waiting time and Turn Around time
avg_wt = wt * 1.0 / NOP;
avg_tat = tat * 1.0 / NOP;
printf("\n\n\nAverage Waiting Time: %f", avg_wt);
printf("\nAverage Turn Around Time: %f\n\n\n", avg_tat);
```

```
    return 0;
}
```

INPUT/OUTPUT