**LOVELY PROFESSIONAL UNIVERSITY**

**(Operating System)**

**Course Code: CSE 316 Course Title:** Operating System

**Name: Bhargava Prabu Reddy M**

**Reg No:** 11808593

**Roll No: 20**

**Section:** K18LC

**Email Address:** Prabu0reddy777@gmail.com

**GitHub Link:** <https://github.com/prabureddy/operating_system_project>

**Problem** **1**:

Consider that a system has P resources of same type. These resources are shared by Q processes time to time. All processes request and release the resources one at a time.Generate a solution to demonstrate that, the system is in safe state when following conditions are satisfied.

Conditions:

1. Maximum resource need of each process is between 1 and P.

2. Summation of all maximum needs is less than P+Q

**Description:**

This problem is based on Round Robin Algorithm. In this problem we are taking input from user and user can enter limited no of jobs.

While solving this first we take value from user and store it in a queue. The value of front and rear used here is -1 as initial they are not at any position. Now process run in queue and if its burst time is less than time given quantum then this process will be completed and we take out that process from queue. If time quantum is less than burst time then the process will again enter in the queue. This will happen till all process are completed.

**Alogithm:**

1. Take process and burst time input from user

for (i = 0; i < n; i++)

{

printf("Burst time P %d : ", i + 1);

scanf("%d", &burst[i]);

process[i] = i;

printf("\n");

}

1. Check in nested loop if burst i is greater than burst j

for (i = 0; i < n; i++)

{

for (j = i; j < n; j++)

{

if (burst[i] > burst[j])

{

temp = burst[i];

burst[i] = burst[j];

burst[j] = temp;

temp = process[i];

process[i] = process[j];

process[j] = temp;

}

}

}

1. for (i = 0; i < n; i++)

{

wait[i] = j;

j = burst[i] + j;

printf("\nP%d : %d\t\t%d", process[i] + 1, wait[i], wait[i] + burst[i]);

avgw += wait[i];

avgt += wait[i] + burst[i];

}

**Complexity:**

Complexity = n log n

**Code Snippet:**

*// main.c*

*#include* <stdio.h>

main()

{

    int n, i, j, avgw = 0;

    float avgt = 0, temp;

    int burst[10], wait[10], process[10];

    printf("\nProcess Scheduling(SJFS) \n");

    printf("\nEnter the number of process : ");

    scanf("%d", &n);

    printf("Enter the Burst time\n");

*for* (i = 0; i < n; i++)

    {

        printf("Burst time P %d : ", i + 1);

        scanf("%d", &burst[i]);

        process[i] = i;

        printf("\n");

    }

*for* (i = 0; i < n; i++)

    {

*for* (j = i; j < n; j++)

        {

*if* (burst[i] > burst[j])

            {

                temp = burst[i];

                burst[i] = burst[j];

                burst[j] = temp;

                temp = process[i];

                process[i] = process[j];

                process[j] = temp;

            }

        }

    }

    printf("\nWaiting time | Turn around time \n");

    j = 0;

*for* (i = 0; i < n; i++)

    {

        wait[i] = j;

        j = burst[i] + j;

        printf("\nP%d : %d\t\t%d", process[i] + 1, wait[i], wait[i] + burst[i]);

        avgw += wait[i];

        avgt += wait[i] + burst[i];

    }

    printf("\n\n");

    printf("Average waiting time : %f\nAverage Turn around time : %f\n\n", avgw /  (float)n, avgt / (float)n);

}

Output:

**Process Scheduling(SJFS)**

**Enter the number of process : 5**

**Enter the Burst time**

**Burst time P 1 : 12**

**Burst time P 2 : 14**

**Burst time P 3 : 3**

**Burst time P 4 : 25**

**Burst time P 5 : 48**

**Waiting time | Turn around time**

**P3 : 0 3**

**P1 : 3 15**

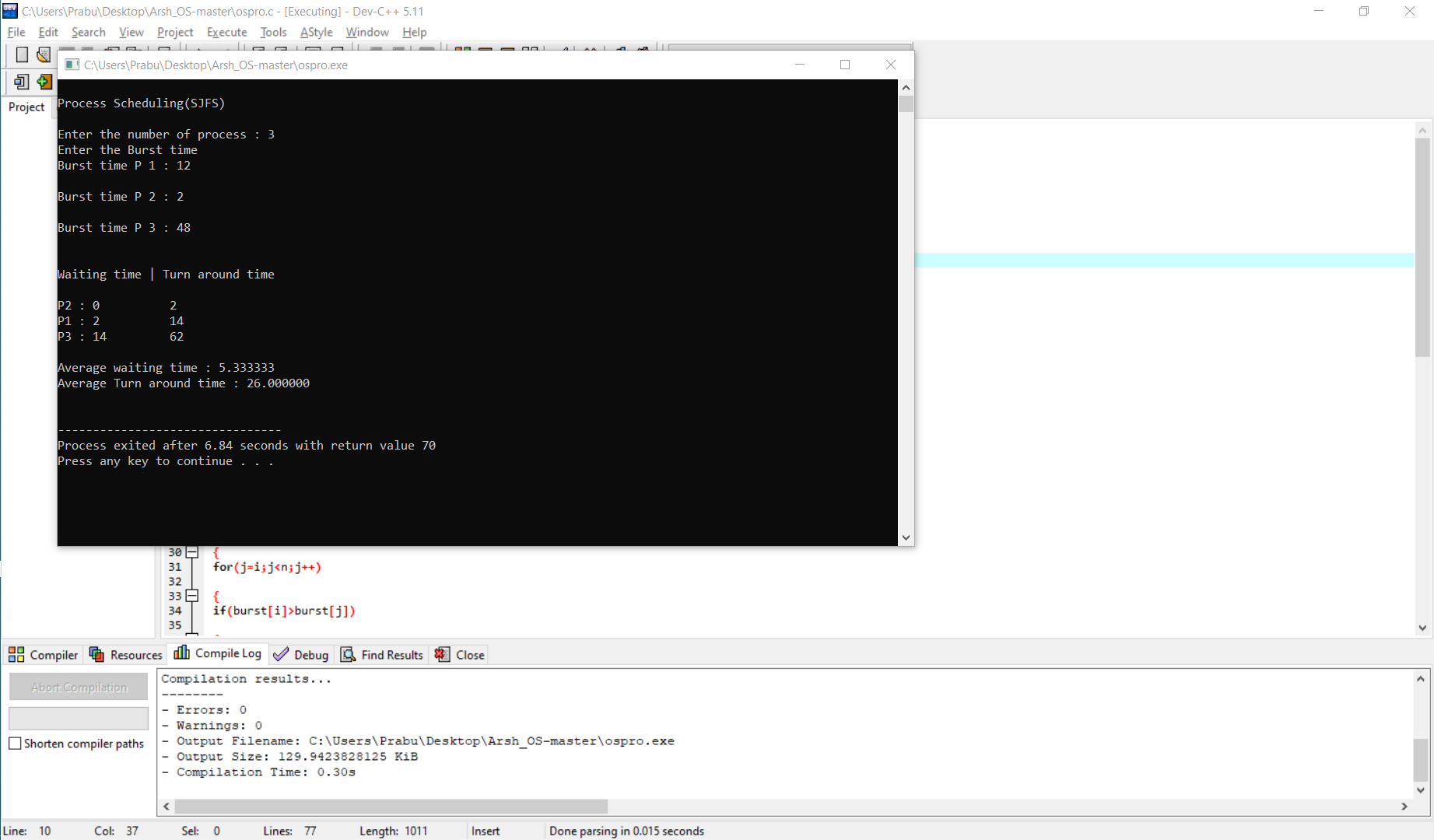
**P2 : 15 29**

**P4 : 29 54**

**P5 : 54 102**

**Average waiting time : 20.200001**

**Average Turn around time : 40.599998**

****