David B.A.De Vieira Velho

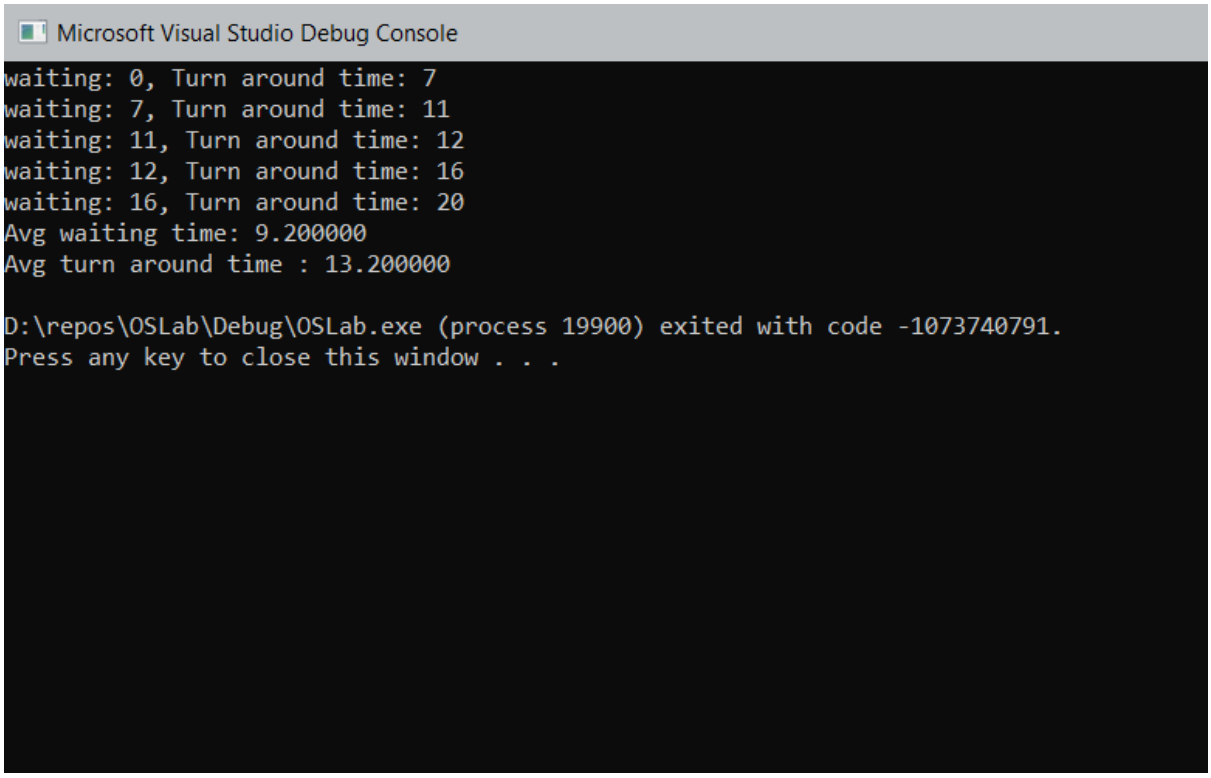
18BCB0142

OS Lab Assignment 2

a) FCFS

```
// OSLab.cpp : This file contains the 'main' function. Program execution begins and ends
there.
//

#include "pch.h"
#include <stdio.h>
#include <stdlib.h>

#define MAX_NO_OF_PROCESSES 5

 typedef struct {
        int processNo;
        int turnAroundTime;
        int burstTime;
        int priority;
        int waitingTime;
}_process;




int main()
{
        int i;
        _process p[4];
        float totalWaitingTime = 0;
        float totalTurnAroundTime = 0;

        p[0].processNo = 1;
        p[1].processNo = 2;
        p[2].processNo = 3;
        p[3].processNo = 4;

        p[0].burstTime = 7;
        p[1].burstTime = 4;
        p[2].burstTime = 1;
        p[3].burstTime = 4;

        //waiting time for first process is 0
        p[0].waitingTime = 0;
```

```
	//FCFS code

	//waiting time of process i+1 will be the sum
	// of previous waiting times of previous processes + their execution times

	for (i = 1; i < MAX_NO_OF_PROCESSES; i++) {
		p[i].waitingTime = p[i - 1].burstTime + p[i-1].waitingTime;
	}

	//to find turn around time
	for (i = 0; i < MAX_NO_OF_PROCESSES; i++) {
		p[i].turnAroundTime = p[i].burstTime + p[i].waitingTime;

	}

	//to find average waiting and turn around time we need waiting time and
turnaroundtime
	for (i = 0; i < MAX_NO_OF_PROCESSES; i++) {
		totalWaitingTime += p[i].waitingTime;
		totalTurnAroundTime += p[i].turnAroundTime;

		printf("waiting: %d, Turn around time: %d\n", p[i].waitingTime,
p[i].turnAroundTime);
	}

	//printing stuff
	printf("Avg waiting time: %f\n", totalWaitingTime /
(float)MAX_NO_OF_PROCESSES);
	printf("Avg turn around time : %f\n", totalTurnAroundTime /
(float)MAX_NO_OF_PROCESSES);

	return 0;
}
```

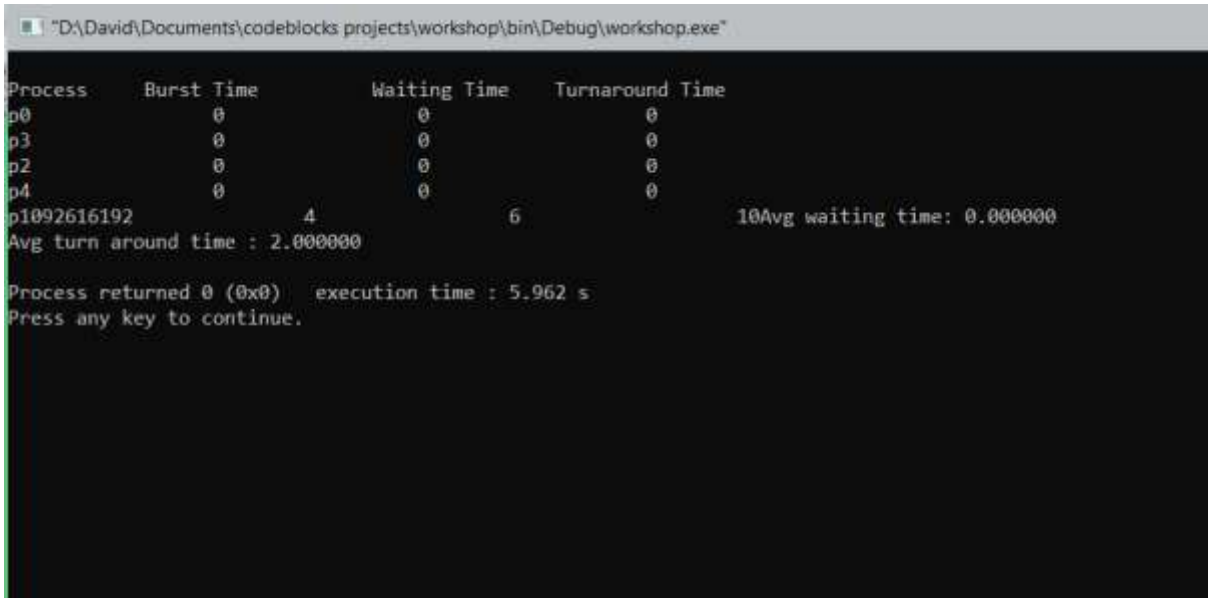


b) In this program , we have to sort it based on the shortest job and arrival time

```
// OSLab.cpp : This file contains the 'main' function. Program execution begins and ends
there.
//


#include <stdio.h>
#include <stdlib.h>

#define MAX_NO_OF_PROCESSES 5

 typedef struct {
        int processNo;
        int turnAroundTime;
        int burstTime;
        int arrivalTime;
        int priority;
        int waitingTime;
}_process;

 void swap(_process *a, _process *b)
 {
         _process temp = *a;
```

```
        *a = *b;
        *b = temp;
}



int main()
{
        int i;
        int j;
        _process p[4];
        float totalWaitingTime = 0;
        float totalTurnAroundTime = 0;

        p[0].processNo = 1;
        p[1].processNo = 2;
        p[2].processNo = 3;
        p[3].processNo = 4;

        p[0].burstTime = 7;
        p[1].burstTime = 4;
        p[2].burstTime = 1;
        p[3].burstTime = 4;


        //arrival time
        p[0].burstTime = 4;
        p[1].burstTime = 5;
        p[2].burstTime = 8;
        p[3].burstTime = 3;


        //arrange based on burst times


        for (i = 0; i < MAX_NO_OF_PROCESSES; i++) {
                for (j = 0; j < MAX_NO_OF_PROCESSES; j++) {

                        if (p[i].burstTime > p[i + 1].burstTime) {
                                swap(&p[i], &p[i + 1]);
                        }

                }
        }

        //waiting time for first process is 0
        p[0].waitingTime = 0;
```

```
	for (i = 1; i < MAX_NO_OF_PROCESSES; i++) {
		p[i].waitingTime = 0;

		for (j = 0; j < i; j++) {
			p[i].waitingTime += p[i].burstTime;
		}

		totalWaitingTime += p[i].waitingTime;
	}



	//turn around time

	printf("\nProcess\t    Burst Time    \tWaiting Time\tTurnaround Time");

	for (i = 0; i < MAX_NO_OF_PROCESSES; i++) {
		p[i].turnAroundTime = p[i].burstTime + p[i].waitingTime;
		totalTurnAroundTime += p[i].turnAroundTime;

		printf("\np%d\t\t  %d\t\t    %d\t\t\t%d", p[i].processNo, p[i].burstTime,
p[i].waitingTime, p[i].turnAroundTime);;
	}

	//printing stuff


	printf("Avg waiting time: %f\n", totalWaitingTime /
(float)MAX_NO_OF_PROCESSES);
	printf("Avg turn around time : %f\n", totalTurnAroundTime /
(float)MAX_NO_OF_PROCESSES);

	return 0;
}
```



c) –
d) Priority

```
// OSLab.cpp : This file contains the 'main' function. Program execution begins and ends
there.
//

#include "pch.h"
#include <stdio.h>
#include <stdlib.h>

#define MAX_NO_OF_PROCESSES 5

 typedef struct {
        int processNo;
        int turnAroundTime;
        int burstTime;
        int arrivalTime;
        int priority;
        int waitingTime;
        int timeRemaining;
}_process;

 void swap(_process *a, _process *b)
 {
        _process temp = *a;
        *a = *b;
        *b = temp;
 }
```

```
int main()
{
	int i;
	int j;
	int time;
	int priority;

	_process p[4];
	float totalWaitingTime = 0;
	float totalTurnAroundTime = 0;

	p[0].processNo = 1;
	p[1].processNo = 2;
	p[2].processNo = 3;
	p[3].processNo = 4;

	p[0].burstTime = 7;
	p[1].burstTime = 4;
	p[2].burstTime = 1;
	p[3].burstTime = 4;


	//arrival time
	p[0].burstTime = 4;
	p[1].burstTime = 5;
	p[2].burstTime = 8;
	p[3].burstTime = 3;

	p[0].priority = 1;
	p[1].priority = 3;
	p[2].priority = 4;
	p[3].priority = 2;

	//priority
	priority = 2;
	//assume priority is some value by input

	printf("\nProcess\t    Burst Time    \tWaiting Time\tTurnaround Time");

	for (i = 0; i < MAX_NO_OF_PROCESSES; i++) {
		if (p[i].priority == priority) {


				for (j = 0; j < MAX_NO_OF_PROCESSES; j++) {
```

```
                                        while (p[j].priority < priority) {
                                                p[i].waitingTime += p[j].burstTime;
                                        }
                                }
                                priority++;

                        }

                        for (j = 0; j < MAX_NO_OF_PROCESSES; j++) {
                                totalWaitingTime += p[i].waitingTime;
                                totalTurnAroundTime = p[i].turnAroundTime;
                        }
                        printf("%d\t%d\t\t%d\t\t%d\t\t%d\t\t%d\n", p[i].processNo, p[i].burstTime,
p[i].priority, p[i].arrivalTime, p[i].waitingTime, p[i].turnAroundTime);
        }


        for (i = 0; i < MAX_NO_OF_PROCESSES; i++) {
                p[i].turnAroundTime = p[i].burstTime + p[i].waitingTime;
                totalTurnAroundTime += p[i].turnAroundTime;

                printf("\np%d\t\t  %d\t\t    %d\t\t\t%d", p[i].processNo, p[i].burstTime,
p[i].waitingTime, p[i].turnAroundTime);;
        }

        //printing stuff


        printf("Avg waiting time: %f\n", totalWaitingTime /
(float)MAX_NO_OF_PROCESSES);
        printf("Avg turn around time : %f\n", totalTurnAroundTime /
(float)MAX_NO_OF_PROCESSES);

        return 0;
}
```