ASSIGNMENT 7

WAP to implement non-preemptive SJF scheduling algorithm (in programming language of your choice). Consider process name, burst time, waiting time and turn around time as the important parameters. Consider arrival time as same for all processes.

Show the output in in tabular format.
| Process Name | Burst Time | Waiting Time | Turnaround Time |

CODE :

```
// C++ program to implement Shortest Job first with Arrival

// Time

#include <iostream>

using namespace std;

int mat[10][6];


void swap(int* a, int* b)

{

	int temp = *a;

	*a = *b;

	*b = temp;

}


void arrangeArrival(int num, int mat[][6])

{

	for (int i = 0; i < num; i++) {

		for (int j = 0; j < num - i - 1; j++) {

			if (mat[j][1] > mat[j + 1][1]) {

				for (int k = 0; k < 5; k++) {

					swap(mat[j][k], mat[j + 1][k]);
```

```
                                    }
                              }
                        }
                  }
            }

void completionTime(int num, int mat[][6])
{
      int temp, val;
      mat[0][3] = mat[0][1] + mat[0][2];
      mat[0][5] = mat[0][3] - mat[0][1];
      mat[0][4] = mat[0][5] - mat[0][2];

      for (int i = 1; i < num; i++) {
            temp = mat[i - 1][3];
            int low = mat[i][2];
            for (int j = i; j < num; j++) {
                  if (temp >= mat[j][1] && low >= mat[j][2]) {
                        low = mat[j][2];
                        val = j;
                  }
            }
            mat[val][3] = temp + mat[val][2];
            mat[val][5] = mat[val][3] - mat[val][1];
            mat[val][4] = mat[val][5] - mat[val][2];
            for (int k = 0; k < 6; k++) {
                  swap(mat[val][k], mat[i][k]);
            }
      }
```

```
}

int main()
{
	int num, temp;

	cout << "Enter number of Process: ";
	cin >> num;

	cout << "...Enter the process ID...\n";
	for (int i = 0; i < num; i++) {
		cout << "...Process " << i + 1 << "...\n";
		cout << "Enter Process Id: ";
		cin >> mat[i][0];
		cout << "Enter Arrival Time: ";
		cin >> mat[i][1];
		cout << "Enter Burst Time: ";
		cin >> mat[i][2];
	}

	cout << "Before Arrange...\n";
	cout << "Process ID\tArrival Time\tBurst Time\n";
	for (int i = 0; i < num; i++) {
		cout << mat[i][0] << "\t\t" << mat[i][1] << "\t\t"
			<< mat[i][2] << "\n";
	}

	arrangeArrival(num, mat);
	completionTime(num, mat);
```

```
        cout << "Final Result...\n";
        cout << "Process ID\tBurst Time\tWaiting "
                "Time\tTurnaround Time\n";
        for (int i = 0; i < num; i++) {
            cout << mat[i][0] << "\t\t"
                << mat[i][2] << "\t\t" << mat[i][4] << "\t\t"
                << mat[i][5] << "\n";
        }
}
```

OUTPUT:

```
Enter number of Process: 4
...Enter the process ID...
...Process 1...
Enter Process Id: 1
Enter Arrival Time: 2
Enter Burst Time: 3
...Process 2...
Enter Process Id: 2
Enter Arrival Time: 0
Enter Burst Time: 4
...Process 3...
Enter Process Id: 3
Enter Arrival Time: 4
Enter Burst Time: 2
...Process 4...
Enter Process Id: 4
Enter Arrival Time: 5

Enter Burst Time: 4

Before Arrange...

Process ID     Arrival Time    Burst Time

1              2               3

2              0               4

3              4               2

4              5               4

Final Result...

Process ID     Burst Time      Waiting Time    Turnaround Time

2              4               0               4

3              2               0               2

1              3               4               7

4              4               4               8
```