FCFS Scheduling

Perumalla Dharan
AP21110010201

(Q) Read the details of a process like a Process ID, and Burst Time and store it in a QUEUE (Implemented using a linked list, the details of each process will be stored in a node in the list). Further, delete the nodes based on FIFO policy from the QUEUE and compute the wait time, completion time, and Turn around time of the processes.

```
// Read the details of a process like a Process ID, and Burst
Time
// and store it in a QUEUE (Implemented using a linked list,
the details
// of each process will be stored in a node in the list).
Further, delete
// the nodes based on FIFO policy from the QUEUE and computed
the wait
// time, completion time, and Turn around time of the
processes.

#include <iostream>
using namespace std;

struct queue
{
    int id, a_time, b_time, w_time, c_time, ta_time;
    queue *next;
};

class FCFS
{
private:
```

```
    queue *front;
    queue *rear;

public:
    FCFS()
    {
        front = NULL;
        rear = NULL;
    }

    bool isEmpty()
    {
        return front == NULL;
    }

    queue addnode(int id, int atime, int btime)
    {
        queue *q = new queue;
        q->id = id;
        q->a_time = atime;
        q->b_time = btime;
        q->c_time = 0;
        q->w_time = 0;
        q->ta_time = 0;
        q->next = NULL;
        if (isEmpty())
        {
            front = q;
            rear = q;
        }
        else
        {
            rear->next = q;
```

```
            rear = q;
        }
    }

    queue deletenode()
    {
        queue queue;
        if (isEmpty())
        {
            cout << "Queue is empty" << endl;
            queue.id = -1;
            return queue;
        }
        else
        {
            queue = *front;
            front = front->next;
            if (front == NULL)
            {
                rear = NULL;
            }
            return queue;
        }
    }
};

int main()
{
    int n;
    cout << "Enter total number of processes" << endl;
    cin >> n;
    FCFS np;
    for (int i = 0; i < n; i++)
```

```
    {
        int id, atime, btime;
        cout << "Enter process id" << endl;
        cin >> id;
        cout << "Enter arrival time" << endl;
        cin >> atime;
        cout << "Enter burst time" << endl;
        cin >> btime;
        np.addnode(id, atime, btime);
    }
    int current_time = 0;
    while (!np.isEmpty())
    {
        queue n = np.deletenode();
        n.w_time = current_time - n.a_time;
        n.c_time = current_time + n.b_time;
        n.ta_time = n.c_time - n.a_time;
        cout << "Process\tWait time\tCompletion time\tTurn
around time" << endl;
        cout << n.id << "\t\t" << n.w_time << "\t\t" <<
n.c_time << "\t\t" << n.ta_time << endl;
        current_time = n.c_time;
    }
    return 0;
}
```

OUTPUT

```
Process Wait time       Completion time Turn around time
1               0               3               3
Process Wait time       Completion time Turn around time
2               1               9               7
Process Wait time       Completion time Turn around time
2               4               18              13
```