Politecnico di Milano
SCUOLA DI INGEGNERIA INDUSTRIALE E DELL'INFORMAZIONE

Advanced Operating Systems
A.A. 2019-2020 – Exam date: February, 06th 2020

Prof. William FORNACIARI

Surname (readable)..... Name (readable).....

Matr..... Signature.....

Q1	Q2	TOT

NOTES

It is forbidden to refer to texts or notes of any kind as well as interact with their neighbors. Anyone found in possession of documents relating to the course, although not directly relevant to the subject of the examination will cancel the test. It is not allowed to leave during the first half hour, the task must still be returned, even if it is withdrawn. The presence of the writing (not delivered) implies the renunciation of any previous ratings.

Question 1 (10 points)

Given the following task set, you are asked to draw the Gantt diagram of the schedule obtained by applying each of the following algorithms:

- Shortest Job First (SJF)
- Shortest Remaing Time First (SRTF)
- Highest Response Ratio Next (HRRN)
- Round-Robin (RR) with the quantum t=2

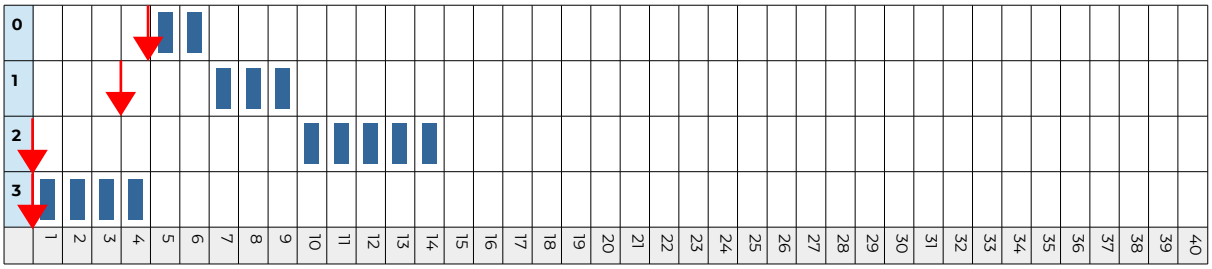
Task	Arrival time (a)	Completion time (C)
0	4	2
1	3	3
2	0	5
3	0	4

Then, for each schedule, compute the average values of waiting time (W), response time (R) and turnaround time (Z).

Finally, make some comments on the performance shown by the different scheduling algorithms, according to the metrics computed above.

1) Shortest Job First (SJF)

Schedule



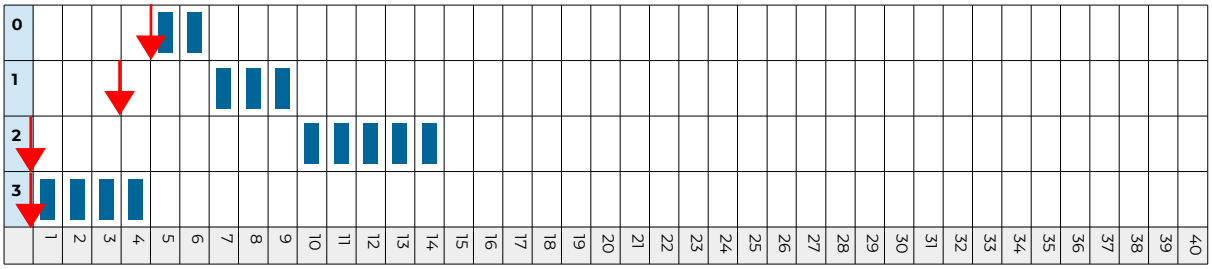
Task	Arrival time	Start time	Finish time	Computation time (C)	Waiting time (W)	Response time (R)	Turnaround time (Z)
0	4	4	6	2	0	2	2
1	3	6	9	3	3	6	6
2	0	9	14	5	9	14	14
3	0	0	4	4	0	4	4

Average values:

```
W = (0 + 3 + 9  + 0) / 4 = 3.00
R = (2 + 6 + 14 + 4) / 4 = 6.50
Z = R
```

2) Shortest Remaining Time First (SRTF)

Schedule



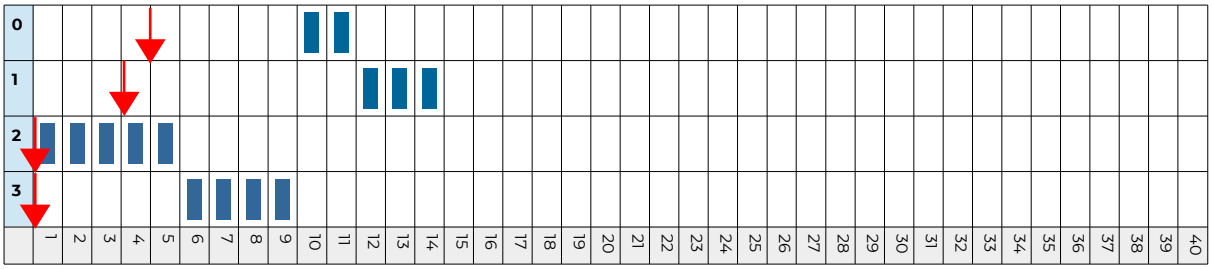
Task	Arrival time	Start time	Finish time	Computation time (C)	Waiting time (W)	Response time (R)	Turnaround time (Z)
0	4	4	6	2	0	2	2
1	3	7	9	3	3	6	6
2	0	9	14	5	9	14	14
3	0	0	4	4 (1)	0	4	4

t0: C_{T2}=5 **C_{T3}=4**
t3: C_{T1}=3 C_{T2}=5 **C_{T3}=1**
t4: **C_{T0}=2** C_{T1}=3 C_{T2}=5
t6: **C_{T1}=3** C_{T2}=5
t6: **C_{T2}=5**

```
W = (0 + 3 +  9 + 0) / 4 = 3.00
R = (2 + 6 + 14 + 4) / 4 = 6.50
Z = R
```

3) Highest Response Ratio Next (HRRN)

Schedule



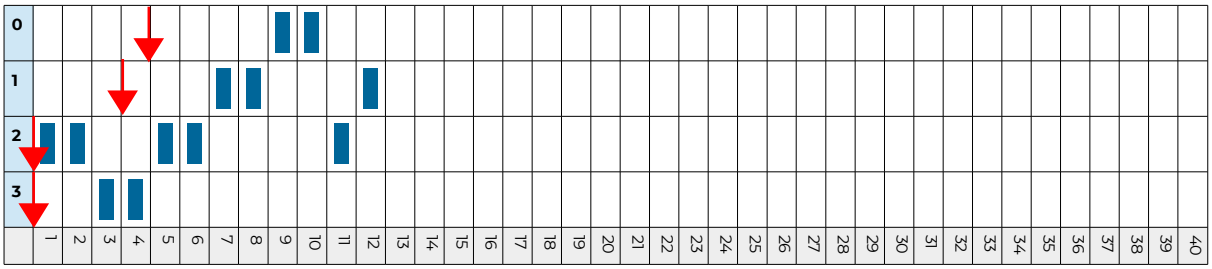
Task	Arrival time	Start time	Finish time	Computation time (C)	Waiting time (W)	Response time (R)	Turnaround time (Z)
0	4	9	11	2	5	7	7
1	3	11	14	3	8	11	11
2	0	0	5	5	0	5	5
3	0	5	9	4	5	9	9

t0: **RR_{T2}=1.00** RR_{T3}=1.00
t5: RR_{T0}=(1+2)/2=1.50 RR_{T1}=(2+3)/3=1.66 **RR_{T3}=(5+4)/4=2.25**
t9: RR_{T0}=(5+2)/2=**3.50** RR_{T1}=(6+3)/3=3.00
t11: RR_{T1}=(8+3)/3=**3.67**

```
W = (5 +  8 + 0 + 5) / 4 = 4.50
R = (7 + 11 + 5 + 9) / 4 = 8.00
Z = R
```

4) Round Robin (RR) t=2

Schedule



Task	Arrival time	Start time	Finish time	Computation time (C)	Waiting time (W)	Response time (R)	Turnaround time (Z)
0	4	10	12	2	6	8	8
1	3	8	14	3	8	7	11
2	0	0	13	5	8	2	13
3	0	2	8	4	4	4	8

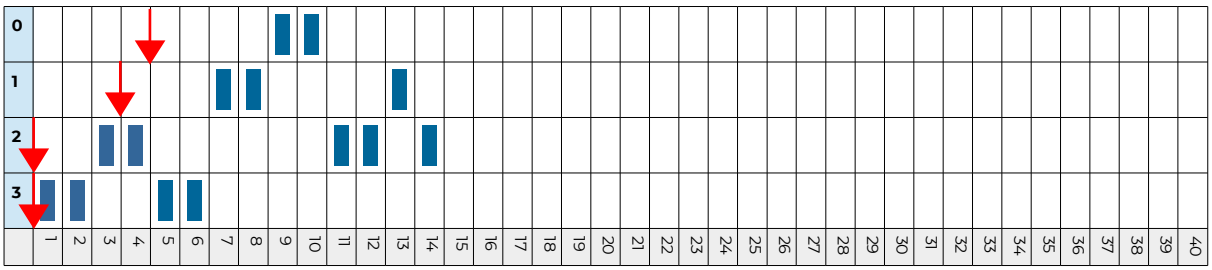
Queue status

t0: **T2**, T3
t2: **T3**, T2
t4: **T2**, T1, T3, T0
t6: **T1**, T0, T3, T2
t8: **T0**, T3, T2, T1

t10: **T3**, T2, T1
t12: **T2**, T1
t13: **T1**

W = (6 + 8 + 8 + 4) / 4 = 6.50
R = (8 + 7 + 2 + 4) / 4 = 5.25
Z = (8 + 11 + 13 + 8) / 4 = 10

Schedule 2 (alternative)



Task	Arrival time	Start time	Finish time	Computation time (C)	Waiting time (W)	Response time (R)	Turnaround time (Z)
0	4	8	10	2	4	6	6
1	3	6	13	3	7	5	10
2	0	2	14	5	9	4	14
3	0	0	6	4	2	2	6

Queue status
t0: **T3**, T2
t2: **T2**, T3
t4: **T3**, T1, T0, T2
t6: **T1**, T0, T2
t8: **T0**, T2, T1
t10: **T2**, T1
t12: **T1**, T2
t13: **T2**

```
W = (4 +  7 +  9 + 2) / 4 = 5.50
R = (6 +  5 +  4 + 2) / 4 = 4.25
Z = (6 + 10 + 14 + 6) / 4 = 9.00
```

Question 2 (13 points)

Implement a C++ template class named SyncPoint with two member functions: **void addData(T data)** and **std::list<T> getData()**. The first member functions must be callable by one or more threads and should put the data item in an ordered queue. The second member function must also be callable by multiple threads and shall return the first **three** data items that have been put in the queue. If the queue contains less than three data items, the function shall wait.

```
//Put here include needed


template<typename T>
class SyncPoint
{
public:
    void addData(T data);

    std::list<T> getData();

    //Complete the rest of the class body and implementation
```

Solution

```
========
#include <list>
#include <mutex>
#include <condition_variable>

template<typename T>
class SyncPoint
{
public:
    void addData(T data);

    std::list<T> getData();

private:
    const int minFill=3;
    std::list<T> queue;
    std::mutex m;
    std::condition_variable cv;
};

template<typename T>
void SyncPoint<T>::addData(T data)
{
    std::unique_lock<std::mutex> lock(m);
    queue.push_back(data);
    if(queue.size() >= minFill) cv.notify_one();
}

template<typename T>
std::list<T> SyncPoint<T>::getData()
{
    std::unique_lock<std::mutex> lock(m);
    while(queue.size() < minFill) cv.wait(lock);
    std::list<T> result;
    for(int i = 0; i < minFill; i++)
```

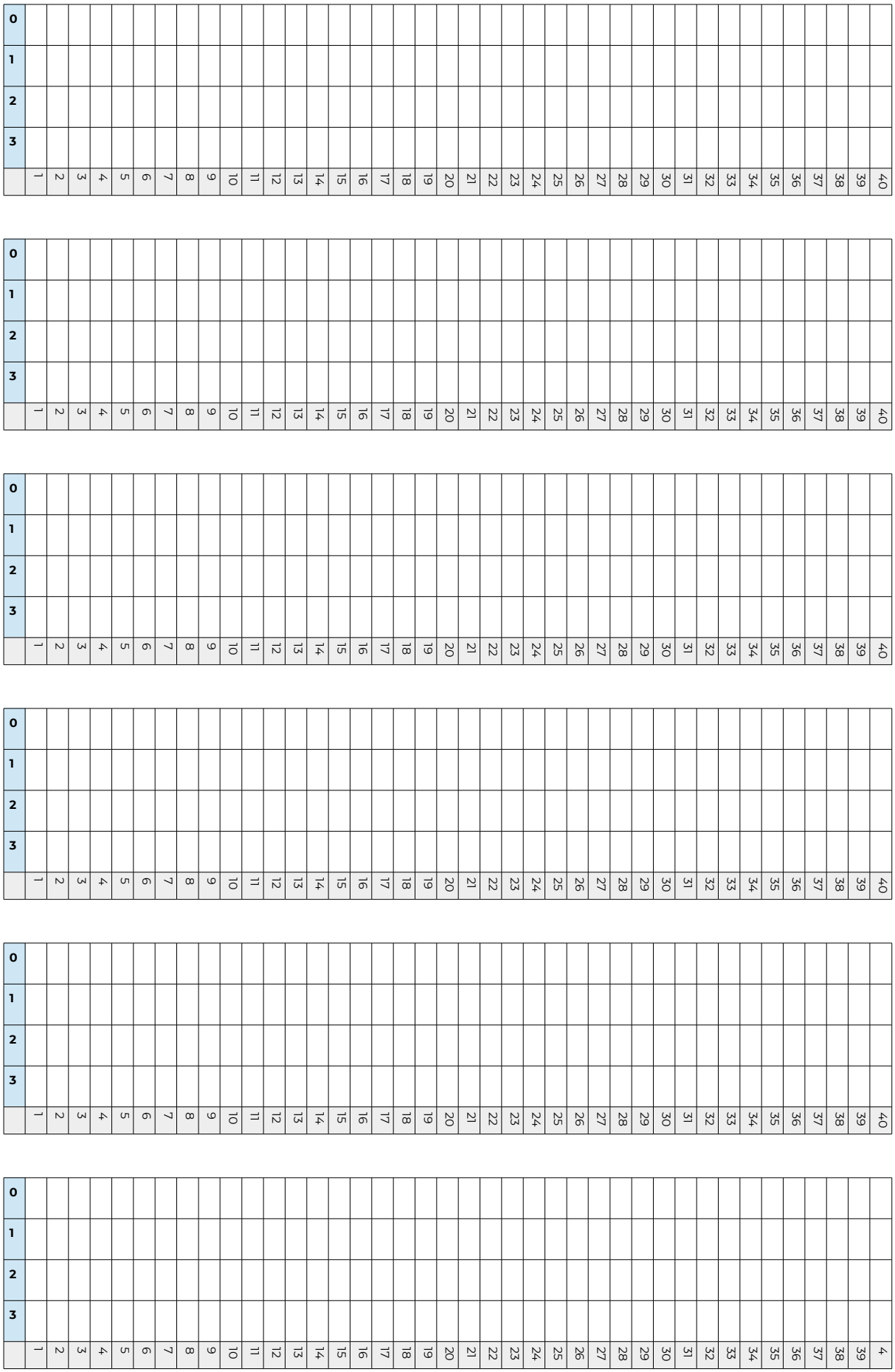
```
    {
        result.push_back(queue.front());
        queue.pop_front();
    }
    return result;
}

// Not part of the solution, just some testing code

#include <iostream>
#include <chrono>
#include <atomic>
#include <thread>

int main()
{
    using namespace std;
    SyncPoint<int> sp;
    sp.addData(1);
    sp.addData(2);
    sp.addData(3);
    sp.addData(4);
    atomic<bool> quit(false);
    thread t([&]{
        while(!quit.load())
        {
            auto r=sp.getData();
            cout<<"size="<<r.size()<<endl;
            for(int i : r) cout<<" "<<i<<endl;
        }
    });
    this_thread::sleep_for(chrono::seconds(1));
    quit.store(true);
    sp.addData(5);
    sp.addData(6);
    t.join();
}
```

Draft pages


0
1
2
3
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40