HALMSTAD UNIVERSITY

Centre for Research on Embedded Systems (CERES)
Embedded Systems Programming
Model Examination, October 13, 2015

Instructions. No reading material, computer or calculator is allowed into the examination; you may only use a paper-based dictionary. The exam comprises 5 questions in 2 pages and will take 3 hours. Before starting to answer the questions, please make sure that your copy is properly printed. Good luck!

Question 1 (20/100 points). Explain how reading from memory differs from reading from memory mapped IO (one difference suffices, (**5 points**), what kind of challenge arises from the differences (mention two challenges, (**10 points**) and how these challenges can be overcome (mention at least one programming technique, (**5 points**).

Question 2 (20/100 points). Consider the following implementation of a program reading a temperature and a pressure sensor, calculating new goal temperature and pressure values based on the values read from the sensors and controlling a thermostat to reach the goal values.

```
int main() {
  int temp, goal_temp;
  int pres, goal_pres;
  while (1) {
     if (New_Temp) {
       temp = Temp_Data;
       calculate_goal_temp(temp, &goal_temp);
     }
     if (New_Pres) {
       pres = Temp_Pres;
       calculate_goal_pres(pres, &goal_pres);
     }
     control_thermostat(goal_temp, goal_pres);
  }
 return ERR_CODE;
}
```

Criticize and explain can go wrong with the above-given program (**10 points**).
Re-write this into a program that does not suffer from the problems you noticed (**10 points**).

Question 3 (40/100 points). Consider the following specification of 3 periodic tasks.

Task	Execution Time	Period = Deadline
A	22	50
B	5	20
C	3	10

3.a. Is this set of tasks schedulable using Rate Monotonic scheduling? Motivate your answer (for your information: $2^{(1/2)} = 1.4$ and $2^{(1/3)} = 1.3$. (**10 points**)

3.b. Show the scheduling of the first instance of A with the first three instances of B and the first 5 instances of C, using both the Rate Monotonic and the Earliest Deadline First algorithm. Assume that the first instance of all three tasks arrive simultaneously. (**15 points**)

3.c. Assume that we modify the task set by adding a new task and also relaxing the assumption of Period = Deadline, as follows.

Task	Execution Time	Period	Deadline
A	22	50	20
B	5	20	15
C	2	10	5
D	3	10	10

Analyze whether this task set is schedulable using deadline monotonic scheduling. (**15 points**)

Question 4 (10/100 points). Assume that you have two tasks Ta_1 and Ta_2 with the periods T_1 and T_2, respectively such that $T_1 < T_2$. Moreover assume that C_1 and C_2 are their worst-case execution times and $D_1 = T_1$ and $D_2 = T_2$ are their deadlines. Show that the worst response time for Ta_2 happens when Ta_1 and Ta_2 arrive at the same time. How many times an instance of Ta_2 should be preempted in such a case? (**10 points**)

Question 5 (10/100 points). Explain how an Android application can spawn a new thread and how the worker thread can interact with the activity.

Answer 1. Reading from memory usually results in reading a useful value stored in the memory location while reading memory-mapped IO does not necessarily result in a useful value. The challenges are:

- Dealing with possibly different rates for different sensor data,
- Not wasting CPU time while waiting for useful data to arrive, and
- Frequently checking for new values while performing time-consuming calculations.

Using interrupt-driven input rather than busy waiting and using concurrent threads for time-consuming calculations are two techniques to overcome these issues.

Answer 2. The problem is that the calculation of the goal values may take long and hence we may miss a new piece of data arriving at a sensor.

To avoid this, we should perform these calculations in concurrent threads. (Of course the following solution has the problem of busy waiting and wasting CPU time, but this is another issue not asked for in this question.)

```
int temp, goal_temp;
int pres, goal_pres;

void temp_sense() {
  while (1) {
      if (New_Temp) {
         temp = Temp_Data;
         calculate_goal_temp(temp, &goal_temp);
      }
    }
  }


void pres_sense() {
  while (1) {
      if (New_Pres) {
        pres = Temp_Pres;
        calculate_goal_pres(pres, &goal_pres);
      }
    }
  }

int main() {
  goal_temp = Default_Temp;
  goal_pres = Default_Pres;
  spawn(temp_sense, 0);
  spawn(pres_sense, 0);
  while (1) {
    control_thermostat(goal_temp, goal_pres);
    }
    return ERR_CODE;
}
```

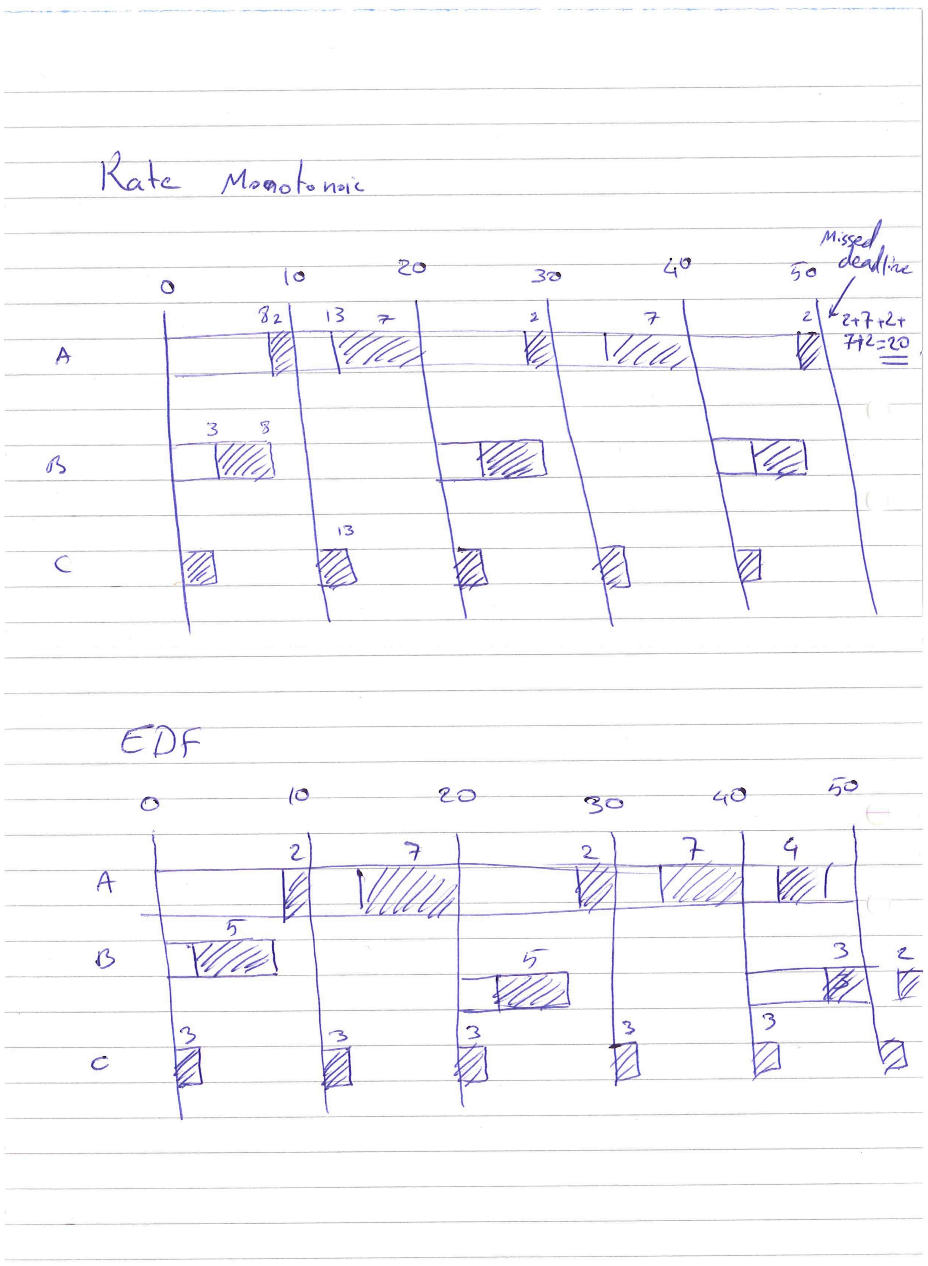
Answer 3.

3.a. The utilization is calculated using the following expression:

U = (22/50) + (5/20) + (3/10) = .99

Since utilization is greater than 3 $(2^{(1/3)}$ -1) = .78, the task set is most likely not schedulable using rate monotonic scheduling.

3.b.



3.c.

The first step is to sort the tasks by their deadlines:

Number	Task	Execution Time	Period	Deadline
1	C	2	10	5
2	D	3	10	10
3	B	5	20	15
4	A	22	50	30

Then, one has to perform the following check for each I <= 4:

$$C_i + \sum_{j=1}^{i-1} \left\lceil \frac{D_i}{T_j} \right\rceil C_j \leq D_i$$

For task 1:

2 + 0 <= 5, pass

For task 2:

3 + [10/10]2 = 5 < = 10, pass

For task 3:

5 + [15/10]3 + [15/10]2 = 5 + 6 + 4 < = 15, pass

For task 4:

22 + [30/20]5 + [30/10]3 + [30/10]2 = 47 < = 30, fail

Hence, the task set is probably not schedulable using deadline monotonic.

Answer 4.

The maximum completion time for Ta2 is T2, when it has to wait till its deadline.
In such a case, |_ T2/ T1 _| instances of Ta1 may preempt the instance of Ta2.

Answer 5. The notation to spawn a new thread is given below where

```
new Thread(new Runnable() {
  public void run() {
     ...
   }
}).start();
```

The worker thread can interact with the main activity by posting a runnable that will be run by the main thread. An example is given below.

```
showtext.post(new Runnable() {
  public void run() {
     showtext.setText("blah blah…”);
   }
 });
```