Department of Computer Engineering, METU

CENG 140

Spring 2024-2025

Take-home exam (THE) 1

Görkem Özer
gorkem@ceng.metu.edu.tr

Due date: 20 April 2025, Sunday, 23:59

1 Part 1 - Assembly-like instruction execution (50 points)

1.1 Task Description

There will be a set of instructions that are simplified versions of the assembly instructions. You are required to have variables that imitate registers, execute instructions to change values in those "registers", and finally print their final states.

Here are the assumptions that you can make:

- There are only 8 registers. Registers hold value of 0 at the beginning.
- There can be at most 3 arguments for instructions.
- There can be at most 20 instructions given in the input.
- Instruction ids start from 1.

Instruction types are:

- **Add:** Takes 2 registers as arguments. Adds the value of the second register to the first one and saves the result to the first register.

  ```
  A 1 5 -> adds the value of R5 to the value of R1 and puts the result to R1
  ```

- **Copy:** Takes 2 registers as arguments. Copies value of the second register and saves it to the first one.

  ```
  C 1 2 -> copies the value of R2 to the value of R1
  ```

- **Decrement:** Takes 1 register as argument. Substracts 1 from the value of the first register and save it to the same register.

  ```
  D 4 -> substracts 1 from the value of R4
  ```

- **Increment:** Takes 1 register as argument. Adds 1 to the value of the first register and save it to the same register.

  ```
  I 8 -> adds 1 from the value of R8
  ```

- **Jump if greater:** Takes 2 registers and 1 instruction id as arguments. Jumps to the instruction with id if the value of the first register is greater than the second register.

  ```
  J 7 2 4 -> jumps to instruction 4 if R7 > R2
  ```

- **Set:** Takes 1 register and 1 integer value as arguments. Puts the given value in the given register.

  ```
  S 1 120 -> sets 120 to the value of R1
  ```

1.2 Input-Output Details

In this THE1, first integer input will always be the task id for both tasks. For the first task:

- Input in the second line will be the number of instructions.
- Remaining lines will be instructions and their arguments.

Here is a sample input for you:

```
1
6
S 8 11
I 2
D 3
D 3
J 3 6 2
A 8 3
```

Explanation:

- Task id is 1. Means the input is for this task.
- There are 6 instructions.
- Instructions are set, increment, decrement, decrement, jump and add.
- Jump instruction is not executed. If it was executed, it would be executed as long as R3 > R6. Each time, it would jump to instruction 2, which is "increment R2".

And the output for the sample input is:

```
R1: 0
R2: 1
R3: -2
R4: 0
R5: 0
R6: 0
R7: 0
R8: 9
```

This is the way to print registers. Even if the register is 0, you should print all of the registers. **Do not forget to put a newline character at the end.**

2 Part 2 - Need for planes (with recursion, 50 points)

2.1 Task Description

Türk Hava Kurumu (Turkish Aeronautical Association) wants you to develop a solution to know the minimum number of planes needed to extinguish a fire on a land.

- THK has different type of planes. Each plane can carry different amounts of water.
- Planes can be effective on a square shaped area.
- Planes can extinguish N x N land where N is a distance in meters.
- Lands are rectangular and have a size of M x N where M and N are distances in meters. M could be equal to N.

You are required to find out the minimum number of planes needed for extinguishing a fire. **ONLY recursive solutions are allowed.**

Here are the assumptions that you can make:

- There could be at most 5 plane types.
- Single unit of land is 10m x 10m. Plane capacities are multiples of 10.
- For effectiveness, only solutions that do not waste water are accepted. For example, using a plane with 480m x 480m capacity on 300m x 300m land is not acceptable.
- Plane capacities are given in descending order.
- **The input cases will allow having at least 1 non-water wasting solution.**

2.2 Input-Output Details

In this THE1, first integer input will always be the task id for both tasks. For the second task:

- Inputs in the second line will contain land length and land width.
- Inputs in the third line will contain number of plane types.
- Remaining lines will tell the capacity of each plane type.

Here is a sample input for you:

```
2
600 900
3
600
300
60
```

Explanation:

- Task id is 2. Means the input is for this task.
- Target land area is 600m x 900m.
- There are 3 plane types.
- Planes have either 600m x 600m , 300m x 300m or 60m x 60m area capacity.

And the output for the sample input is:

```
3 planes
```

Because you would send 1 plane with 600m x 600m capacity and 2 planes with 300m x 300m capacity. **Do not forget to put a newline character at the end.**

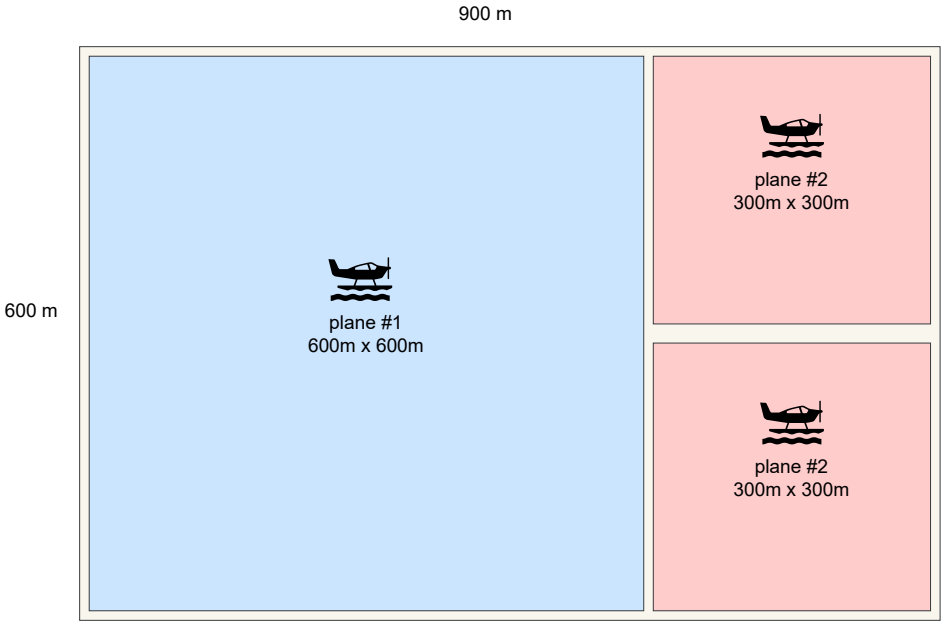

Figure 1: Solution for sample question in task 2

3 Regulations

- **Programming Language:** C
- **Libraries and Language Elements:**
  You can only use these libraries: **stdio.h, limits.h**. You can use conditional clauses (switch/if/else if/else), loops (for/while), arrays. **You can NOT use any further elements beyond those.** You can define helper functions.
- **Compiling and running:**
  You should be able to compile your codes and run your program with following commands:

  ```
  >_ gcc the1.c -ansi -pedantic-errors -Wall -o the1
  >_ ./the1
  ```

  If you are working with ineks or you are working on Ubuntu OS, you can feed your program with input files instead of typing inputs. This is a feature of Ubuntu OS and an equivalent of typing inputs. This way you will test your program faster:

  ```
  >_ ./the1 < inp1.txt
  >_ ./the1 < inp2.txt
  ```

- **Submission:**
  You should submit your the1.c implementation. Late submission IS NOT allowed, it is not possible to extend the deadline and **please do not ask for any deadline extensions**.
- **Evaluation:**
  Your codes will be evaluated based on several inputs including, but not limited to the test cases given to you as example. If your program gives correct outputs for all cases, you will get 100 points.
- **Cheating:**
  We have zero tolerance policy for cheating. People involved in cheating will be punished according to the university regulations and will get 0. Sharing code between each other or using third party code is strictly forbidden. Even if you take a "part" of the code from somewhere/somebody else - this is also cheating. Please be aware that there are "very advanced tools" that detect if two codes are similar. So please do not think you can get away with by changing a code obtained from another source.