Lab #1. Warm-Up Exercise

Prof. Jaeseung Choi

Dept. of Computer Science and Engineering

Sogang University

About the Labs

- **Labs will count for 40% of the total score in semester**
- **We will have four lab assignments (tentative)**
  - **Lab #1:** Warm-up Exercise (4%)
  - **Lab #2:** Interpreter for Imperative Language (8%)
  - **Lab #3:** Interpreter for Functional Language (8%)
  - **Lab #4:** Type Inference (20%)
- **Today: Lab #1. Warm-up Exercise**
  - Simple F# programming problems
  - Carefully read the previous F# tutorial – that would be enough
  - Another goal is to get familiar with the skeleton code structure
- **The lab assignments will get harder step by step**
  - So don’t give up the lab too early (Lab #1 is almost a *gift*)

General Information

- **Check the *Assignment* tab of *Cyber Campus***
  - Skeleton code (**`Lab1.tgz`**) is attached together with this slide
  - Submission will be accepted in the same post, too
- **Deadline: 3/28 Thursday 23:59**
  - Late submission deadline: **3/30** Saturday 23:59 **(-20% penalty)**
  - Delay penalty is applied uniformly **(not problem by problem)**
- **Please read the instructions in this slide carefully**
  - This slide is a step-by-step tutorial for the lab
  - It also contains important submission guidelines
    - If you do not follow the guidelines, **you will get penalty**

Skeleton Code Structure

- **Copy `Lab1.tgz` into CSPRO server and decompress it**
  - This course will use cspro**2**.sogang.ac.kr (**don't miss the 2**)
  - Don't decompress-and-copy; copy-and-decompress
- **`P1/~P7/`: Directory for each problem**
- **`check.py`: Script for self-grading (explained later)**
- **`config`: Used by the grading script (you may ignore)**

```
jschoi@cspro2:~$ tar -xzf Lab1.tgz
jschoi@cspro2:~$ cd Lab1/
jschoi@cspro2:~/Lab1$ ls
check.py  config  P1  P2  P3  P4  P5  P6  P7
```

Problem Directory Structure

- **Each directory will contain three files**
  1. **`P*.fs`**: Source file that contains the function you must fill in
     - **The only file that you have to fix and submit**
  2. **`Main.fs`**: source file that contains the test code for your code
     - Test cases are embedded within this file
  3. **`P*.fsproj`**: Project information file
     - Internally used by the **`dotnet`** command; you may ignore

```
jschoi@cspro2:~/Lab1$ cd P1/
jschoi@cspro2:~/Lab1/P1$ ls
Main.fs  P1.fs  P1.fsproj
```

Problem Specification

- **The requirement of each function that you have to implement is given in the comment above the function**
  - Ask a question if you need a clarification of the specification
  - The examples (test cases) in **`Main.fs`** may also help your understanding: let's take a look at them in the next page

`P1/P1.fs`

```
...

/// Return a list reversed from the argument 'l'.
let rec reverse (l: List<'a>) : List<'a> =
  [] // TODO
```

Test Code (Test Cases)

- **In general, `Main.fs` will have a structure like below**
  - Ex) **`r1`** will be **`"O"`** if **`reverse [1; 2; 3]`** returns **`[3; 2; 1]`**

P1/Main.fs

```
let test inp ans =
  try if reverse inp = ans then "O" else "X" with _ -> "E"

let r1 = test [1; 2; 3] [3; 2; 1]
...
```

- **You can build and run `P1.fs` + `Main.fs` as shown below**
  - As we have learned in the previous F# tutorial

```
jschoi@cspro2:~/Lab1/P1$ dotnet build -o out
jschoi@cspro2:~/Lab1/P1$ ./out/P1
XXXXX
```

Constraints

- **For some problems, certain constraints are given in the comment of the source file**
  - Ex) Do not use certain built-in library functions
  - Ex) Do not fix the provided type definition
- **Ensure that your code satisfies these constraints**
  - **You will lost the whole point if you violate the constraints**

P1/P1.fs

```
// (Note) In this problem, you are NOT allowed to use the
// pre-defined library function, 'List.rev'.

/// Return a list reversed from the argument 'l'.
let rec reverse (l: List<'a>) : List<'a> =
...
```

Self-Grading Script

- **If you think you have solved all the problems, you can run `check.py` as a final check**
  - `'O'`: Correct, `'X'`: Incorrect, `'E'`: exception, `'C'`: Compile error, `'T'`: Timeout (maybe infinite recursion)

```
jschoi@cspro2:~/Lab1$ ./check.py
[*] P1: OOOOO
[*] P2: OOOOO
[*] P3: OOOOO
[*] P4: XXXXX
[*] P5: XXXXX
[*] P6: XXXXX
[*] P7: XXXXX
```

Test Cases for Grading

- **I will use different test case set to grade your code**
  - This means even if you pass all the provided test cases, it does not guarantee that you will get 100 pt.
- **So you are encouraged to test your own code with other various inputs**
- **Some students ask me to provide more test cases, but it is important to practice this on your own**

Hints

- **In P5, you may have to use *recursion* in a different style**
  - Review the **`listSum2`** function in the F# tutorial slide

```
let rec sumHelper acc lst =
  match lst with
  | [] -> acc
  | head :: tail -> sumHelper (acc + head) tail

let listSum2 lst = sumHelper 0 lst
```

- **In P6, there are several possible approaches**
  - For example, you can use the functions in **`List`** or **`Map`** module
  - It is also possible to solve the problem without using any pre-defined functions in the library

Problem Information

- **Seven problems in total**
  - **From P1 to P4:** 10 point each (relatively easy problems)
  - **From P5 to P7:** 20 point each
  - 100 point in total
- **You will get the point for each problem based on the number of test cases that your code passes**

Submission Guideline

- **You should submit seven F# source code files**
  - **`P1.fs`**
  - **`P2.fs`**
  - **`...`**
  - **`P7.fs`**
- **If the submitted file does not compile by typing "`dotnet build`", cannot give you any point for that problem**
- **Submission format**
  - Upload these files directly to *Cyber Campus* (**do not zip them**)
  - **Do not change the file name** (e.g., adding any prefix or suffix)
  - If your submission format is wrong, you will get **-20% penalty**