

Programming Assignment #2.

F# Exercise

Prof. Jaeseung Choi

Dept. of Computer Science and Engineering

Sogang University

Before We Start

- **Once you get familiar to the F# language, these exercise problems will be really easy**
 - Each of the problem can be solved with just few lines of code
 - Carefully read the previous **F# tutorial** – that would be enough
- **The assignment and project will get harder step by step**
 - So don't give up too early (this assignment is almost a *gift*)
- **(Remind) You can get the help from ChatGPT**
 - But first try to solve the problems by yourself
 - Ask ChatGPT only after that, when you don't have a clue
 - If you don't practice programming at this stage, you will have a big trouble in the later course project

General Information

- **Check the "Assignment #2" post in *Cyber Campus***
 - Skeleton code (HW2.tgz) is attached in the post
 - Deadline: **11/13 Wednesday 23:59**
 - Submission will be accepted in that post, too
 - Late submission deadline: **11/15 Friday 23:59 (-20% penalty)**
 - Delay penalty is applied uniformly (**not problem by problem**)
- **Please read the instructions in this slide carefully**
 - It also contains important submission guidelines
 - If you do not follow the guidelines, **you will get penalty**

Skeleton Code Structure

- Copy HW2.tgz into CSPRO server and decompress it
 - You must connect to csprou2.sogang.ac.kr (N = 2, 3, or 7)
 - Don't decompress-and-copy; copy-and-decompress
- P1~P7: Directory for each problem
- check.py: Script for self-grading
- config: Used by the grading script

```
jschoi@csprou2:~$ tar -xzf HW2.tgz
jschoi@csprou2:~$ cd HW2/
jschoi@csprou2:~/HW2$ 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:~/HW2$ cd P1/  
jschoi@cspro2:~/HW2/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 "0" if reverse [1; 2; 3] returns [3; 2; 1]

```
P1/Main.fs
let test inp ans =
    try if reverse inp = ans then "0" 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:~/HW2/P1$ dotnet build -o out
jschoi@cspro2:~/HW2/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> =  
...
```


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

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@csp2:~/HW2$ ./check.py
[*] P1: 00000
[*] P2: 00000
[*] P3: 00000
[*] P4: XXXXX
[*] P5: XXXXX
[*] P6: XXXXX
[*] P7: XXXXX
```

Problem Information

- **Seven problems, 100 point in total**
 - From P1 to P4: 10 point each
 - From P5 to P7: 20 point each
- **You will get the point for each problem based on the number of test cases that you pass**
 - If your code does not compile with "`dotnet build`" command, **cannot give you any point** for that problem

Submission Guideline

- You should submit seven F# source code files
 - P1.fs
 - P2.fs
 - ...
 - P7.fs
- No report is required for this assignment
- 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**