Lab 1

Due Wednesday, February 12 by midnight

Handout 3
CSCI 334: Spring 2025

Coding Guidelines

Each question in this assignment should go into the appropriate project directory. For example, the solution to question 1 should be in a folder called "`q1`". When a solution is a program, one should be able to `cd` into the question directory and then run your program by typing the command "`dotnet run`", with additional arguments depending on the question.

Every program should be split into two pieces: a "`Program.fs`" file that contains the `main` method and associated program-startup helpers (if needed), and another "`Library.fs`" file that contains the function(s) of interest in the question. Library code should be contained within a module named "`CS334`". Be sure to provide usage output (defined in `main`) for all programs that require arguments. For full credit, your program should both build and run correctly.

If any of your programs take input from the user, be sure that your program validates input: when a user fails to supply input, or supplies input that does not make sense, your program should print a usage message and return with a nonzero exit code. Users should never experience a program crash in this class; exceptions should be prevented from arising or be caught whenever bad input is encountered. Think through problem corner cases carefully.

Turn-In Instructions

Turn in your work using your assigned `git` repository. The name of your repository will have the form `https://aslan.barowy.net/cs334-s25/cs334-lab01-<USERNAME>.git`. For example, if your CS username is `abc1`, the repository would be `https://aslan.barowy.net/cs334-s25/cs334-lab01-abc1.git`.

You should have received an invite to commit to the repository via email. If you did not receive an email, please contact me right away!

Single-Author Programming Assignment

This is a solo lab. You may work with another classmate to understand what the problems ask, but you are not permitted to develop solutions together. Submitted solutions must be exclusively your own. Please refer to the section "single author programming assignments" in the honor code handout for additional information. You do not need to submit a `collaborators.txt` file for this assignment. You are always welcome to ask me for clarification if the above is unclear in some circumstance.

This assignment is due on Wednesday, February 12 by midnight.

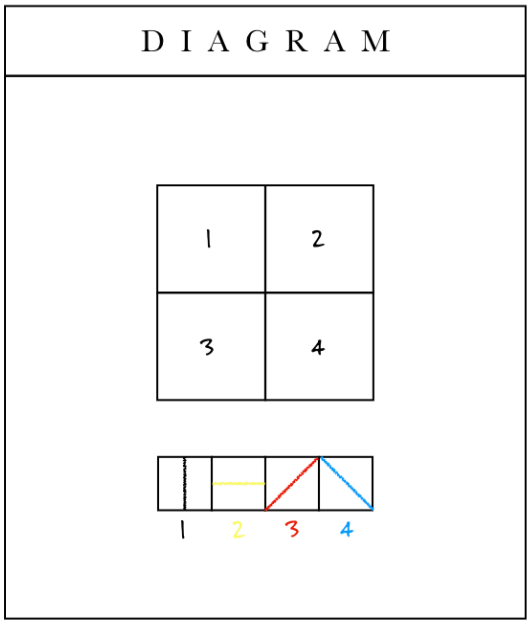
Reading

1. **(If Needed)** "Getting started with the class git accounts" short video at `https://youtu.be/NLcFlXSEGPA`
2. **(Required)** "Preface"
3. **(Required)** "What is a Programming Language?"
4. **(Required)** "An Introduction to F#"
5. **(Required)** "More F#"

Problems

Q1. (25 points) ..... Is this a program?

Run the following "program."




```
A square divided horizontally and vertically
into four equal parts, each with a different
color and line direction.

Red, yellow, blue, black pencil
```

Take a photo of the result with your phone and add it to your submission. **Put your solution in a directory named "`q1`"**. Be sure to name the image file `lewitt.png` (or `lewitt.jpg`, etc).

Q2. (25 points) ..... Hello Favorite Band

Write an F# program that prints your name and your current favorite music act/band. For example, my program would print:

```
Dan Barowy
Underworld
```

One should be able to run your program on the command line like so.

```
$ dotnet run
Dan Barowy
Underworld
```

Remember that F# always expects your `main` function to return an `int`, so your program should probably return zero.

Put your solution in a directory named "`q2`". Be sure to name the program file `Program.fs`.

Q3. (20 points) ..... F# Types

Define the following functions as given by their type signatures below. Be sure to put them in a `Library.fs` file in a module called `CS334`. Your `main` function in `Program.fs` should call each of these functions with arguments of your choosing, printing out the result using `printfn`.

(a) `mean: float -> float -> float`
`mean` should compute the arithmetic mean of two floating point arguments.

(b) `meanOfPair: float * float -> float`
`meanOfPair` should compute the arithmetic mean of a pair of floating point numbers.

(c) `runIt: ('a -> 'b) -> 'a -> 'b`
`runIt` should take two arguments, a function and an input to that function, then run the function on the given input, returning the output.

(d) `runItTwice: ('a -> 'a) -> 'a -> 'a`
`runItTwice` should take two arguments, a function and an input to that function. It should run the function on the input, then run the function again on the output, returning the second output.

(e) `giveItBackTwice: 'a -> 'a * 'a`
`giveItBackTwice` should return a pair of whatever it is given.

Arithmetic expressions in F# look like arithmetic in Java. `2 + 2` or `2 * 2` are valid F# expressions. A type annotation containing a `*` is a tuple. For example, a `int * int` like `(1,2)` is a pair of integers; a `int * string * char` like `(7,"eight",'9')` is a three-tuple containing an integer, a string, and a character. A type containing a `->` is a function and types beginning with a `'` are polymorphic; see the reading for details on both of these.

The project directory for this question should be called "`q3`". Because all of your functions will be called only by you there is no need to validate input for this question.

Q4. (15 points) ..... Conditional expressions

In F#, a conditional expression looks like the following:

```
if x = 0 then
  printfn "It's zero!"
else
  printfn "It's not zero!"
```

Equality tests are written using `=`. F# does not need to use `==` to distinguish between equality and assignment because its syntax ensures that the meaning always clear.

Like everything else in a functional language, conditionals are expressions, which means that they return values. This becomes clearer when we write them inline.

```
let message = if x = 0 then "It's zero!" else "It's not zero!"
printfn "%s" message
```

Write the function,

```
flip: unit -> string
```

that returns either `"heads"` or `"tails"` by sampling a random integer between 0 and 1 inclusive.

You will need to use a random number generator to solve this problem. The following code snippet creates a random number generator and calls its `Next(n: int)` method, which samples a random `int` between `0` inclusive and `n` exclusive.

```
let r = System.Random()
let num = r.Next 2
```

You should be able to run your program on the command line like so.

```
$ dotnet run
tails
```

Remember that F# always expects your `main` function to return an `int`.

The project directory for this question should be called "`q4`". Don't forget to put the `flip` function in the appropriate location.

Q5. (15 points) ..... Working with arrays

In F#, arrays can be created in a number of ways. Here, we show the two simplest ways. First, one can create an array literal. For example, here we create an array literal with five elements in the `fsharpi` REPL:

```
> let arr = [| 1; 2; 3; 4; 5 |];;
```

Note that we terminate the expression above with a `;;` to let `fsharpi` know that we have completed typing our expression. When writing code outside of `fsharpi`, you do not need the `;;` terminator. `fsharpi` prints the following, to let us know how it evaluated what we wrote:

```
val arr : int [] = [|1; 2; 3; 4; 5|]
```

The second way to create an array is to use the `Array.zeroCreate` constructor. This function creates an array of length n, filled with the "default" value for the given type. For example, the "default" value for an `int` is `0`. The "default" value for a `string` is `null`. For this reason, we need to supply a type annotation, otherwise F# does not know which default value to use.

```
> let arr1: int[] = Array.zeroCreate 10;;
val arr1 : int [] = [|0; 0; 0; 0; 0; 0; 0; 0; 0; 0|]

> let arr2: string[] = Array.zeroCreate 7;;
val arr2 : string [] = [|null; null; null; null; null; null; null|]
```

We can access an element of an array with array index notation, `[n]`. For example,

```
> let arr = [| 1; 2; 3; 4; 5 |];;
val arr : int [] = [|1; 2; 3; 4; 5|]

> arr[3];;
val it : int = 4
```

For this question, write the function `nth` that returns the n^{th} element in an array of strings. In other words, write a function:

```
nth: int -> string[] -> string
```

You should be able to run the program on the command line as follows:

```
$ dotnet run 6 the mountains the mountains we greet them with a song
greet
```

Here are some tips to help you with this problem.

(a) The `args` array has type `string[]`. You will need to convert the first element—`6` in the example above—into an `int`. A `string` can be converted to `int` using the `int` function, like so:

```
> let i = int "4";;
val i : int = 4
```

(b) The `main` function is expected to return an `int`, which is the exit code. However, you can also exit with a given exit code at any point in the program by using the `exit` function.

```
exit 1
```

(c) You can print using the `printfn` function. For example, `printfn "%s" "hi"` prints `hi` on the console, while `printfn "%d" 4` prints `4`.

(d) Your program should produce output like the following when given bad input.

```
$ dotnet run 2
Usage: dotnet run <n> <arg_1> .. <arg_n>
```

Since we have not yet discussed exception handling in F#, you may assume that when the user supplies a first argument, that string is always numeric.

(e) As in Python, arrays can be sliced in F#. For example, here we take the slice of the following array from indices 3 to 4 inclusive.

```
> let a = [|1;2;3;4;5;6|];;
val a: int array = [|1; 2; 3; 4; 5; 6|]

> a[3..4];;
val it: int array = [|4; 5|]
```

(f) The function `Array.length: 'a[] -> int` computes the length of an array.

The project directory for this question should be called "`q5`". You should be able to run your program on the command line by typing, for example, "`dotnet run 1 hello world`".

Q6. ($\frac{1}{10}^{\text{th}}$ bonus point) ..... Optional: Feedback

I always appreciate hearing back about how easy or difficult an assignment is.

For $\frac{1}{10}^{\text{th}}$ of a bonus to your final grade, please fill out the following Google Form.