**Names:** Ryan Driscoll, Thom Lippincott, Jacob B. West

**Language:** F#

**Coding environment**: <version, OS, compiler used, **and URL if applicable**>

Visual Studio 2019 on a Windows Machine using .NET core 3.1

### **Distinct pieces of functionality:**

1. A user can create a bank account and log in: Account.fs
2. A user can delete an account: Account.fs, Bank.fs
3. A user can check their balance: Account.fs
4. A user can deposit money and withdraw money: Bank.fs, Account.fs
5. A user can transfer money from one account to another: Bank.fs, Account.fs
6. A user can use different forms of payment to deposit money: <Insert file names here>

### **Section 1: Comparison to C++, Java, Python, and/or Scala**

1. **What is the language’s philosophy?**

F# is a functional language managed by Microsoft. According to Microsoft’s documentation the language is data-oriented and the code commonly involves transforming data with functions (i.e. functional programming). Therefore when looking at other languages such as C++, Java, Python, and Scala, F# is most related to Scala due to the functional approach of solving problems.

1. **Compare and contrast your language in terms of the location it is used.**

Common places F# is used in industry is for machine learning and web development. The language has not gained the popularity such as Python for machine learning or javascript or C# for web development, however it is still used in these areas, specifically when data integrity is of concern.

1. **Compare and contrast your language in terms of where it excels and where it fails**
   * **Excels**

F# excels in a few areas. First because it is a functional language and focuses on transforming data, it is strong is the areas of data science and applications where data integrity is of essence. Because F# promotes immutable objects as the default, it reduces unwanted side effects from interactions within the code. Lastly, F# provides first class functions and powerful pattern matching. You can also group data into records, making the data manipulation easy.

* + **Fails**

F# is not as portable as other languages as it must run in the .NET environment or in a browser. There also is not overloading of functions (this makes sense since it is functional language), which in turn does not allow a developer to have the same name in a module, making it difficult to name functions. Data structures and objects can also get very complex very fast due to the immutability of them.

1. **Compare and contrast your language in terms of Portability, Simplicity, Orthogonality, AND Reliability.**

<Note: maximum of one page, and subheading are suggested>

* + **Portability**

F# can run in two separate environments. The first is within the .NET ecosystem and the second is within a browser, specifically through a few tools such as Binder and Fable which were researched on the web, however these are smaller tools used more for prototyping and not full-scale production applications. Because of the limited areas in which F# can run, it is not as portable as other languages such as Java and C, which are intended to run everywhere.

* + **Simplicity**

F# does excel in the realm of simplicity and finds a good balance in the syntax being easy to write and understand, but not so simple that context gets lost in the syntax. The language prides itself in a lightweight syntax that once, learned, reads very similar to English. Because it is a weakly typed language, data types such as int, and string can be replaced with let, which makes the language more readable. The same can be said for the syntax in branching structures. A simple if statement reads “if (condition) then (do stuff)”. This again reads like the English language and results in clean, easy to understand code just by utilizing small English keywords.

* + **Orthogonality**

<answer here>

* + **Reliability**

<answer here>

### **Section 2: Syntax, OOP**

1. **A) Write an example of one type of assignment expression in the language.**

let myBalance = currentBalance – withdrawAmount

**B) Then write the EBNF for a generic version of this assignment expression with all tokens defined.**

Var <number> 🡪 0..9

var<floatNumber> 🡪 {<number>}.{<number>}

Var <equals> 🡪 <floatNumber> = <floatNumber>

1. **A) How does the language support extension etc. (single inheritance, interfaces, root object, class OOP, prototype OOP, other OOP, file importing, file extension, plugins, piping, module linking, etc.)?   
   <**answer here>

**B) Give an example.**

<answer here. Note: Unless it is esoteric, there will be some extensibility. If you say it does not support class-based OOP (the variety used almost exclusively in this class), so there is no extensibility, I will not believe you as that mean everything must be done in one file! Bash supports extensions! Max at 2 options.>

1. **A) How does the language handle module/namespace/packages/etc..**

F# has namespaces and modules in order to organize code into similar related areas based on functionality. Declaring a namespace occurs at the top of the file and every piece of code written after the declaration is considered to be in that namespace. Namespaces cannot have similar names when creating a new one. A function or value cannot be directly within a namespace, which the introduces the idea of modules.

Modules are considered to be the next level down in the organizational hierarchy. Modules will contain functions, values, types, etc. A module is a way to group code that has similar functionality and a namespace can have multiple modules within it. Within an F# program a developer can reference different namespaces and modules by using the open keyword, which is the equivalent to say an import keyword in C++ or Java. The referenced namespace or module must be in the same project or be included at the assembly level.

**B) What is the scope operator(s)? Alternatively, how to pick which variable if two code courses contain the same name?**

<answer here. Note: the scope operator in C++ is ::. In Java, it is . Python has a few options. >

**4.** **A) Does the language allow function overloading (name repetition), function redefinition, and/or function overriding?**F# does not allow function overloading (with name repetition). This makes sense since F# is a functional language, allows first class functions and commonly assigns a function to a variable. Because of the default immutability you cannot redefine a function later on and/or override a function. Double check me on this?

**B) Give example syntax if it does.**

<answer here. Note: redefinition and overriding are NOT the same thing. >

### **Section 3: Binding, Type system, and data type range**

1. **Is the language static or dynamically typed? Give example syntax in code.**

In the Microsoft documentation F# is considered to be a static typed language. This might seem strange since the syntax implies otherwise, however behind the scenes at compilation the compiler will determine the type of every variable and if the compiler cannot determine type the developer must apply a type in the code in order to have successful compilation.

Not sure if this is the case but it is in the Microsoft docs

1. **Is the language static or dynamically scoped? Give an example in code.**

<answer here>

1. **How is the language read (in-fix, pre-fix, a mix, etc)? Give an example in code or in a diagram. (max of 3 examples).**

<answer here>

1. **What are the built-in data types and their ranges? (list 4-10, or send me a note if you believe that are less than 4)**

|  |  |
| --- | --- |
| **Data Type** | **Range** |
| **byte** | **0 to 255** |
| **int** | **-2,147,483,648 to 2,147,483,647** |
| **single** | **+- 1.5e-45 to +- 3.4e38** |
| **double** | **+- 5.0e-324 to +- 1.7e308** |
| **decimal** | **+- 1.0e-28 to +- 7.9e28** |
| **Char** | **U+0000 to U+ffff** |
| **String** | **0 to approximately 2 billion characters** |
| **bool** | **true or false** |

### **Section 4: Control flow, Function, specialties**

1. **What are the selection and repetition structures of the language, and what is their syntax?**

F# contains if, elif and else statements for branching. The keyword then is used if a condition in an if or elif statement is true. A simple example is shown below:

Insert example

There are 3 types of basic loops in F# used for repetition. They are the for in loop, the for to loop and the while do loop. The for in loop is similar to a foreach loop say in C# or other languages. The for to loop is the equivalent to a normal for loop in other languages. Lastly, the while do loop in F#’s syntax for a while loop. Examples of each loop are shown below:

Insert examples

1. **Are functions pass-by-value, pass-by reference, etc.? Give example syntax in code.**

<answer here>

1. **Describe at least two of the language specialties.**

Specialty one:

<answer here>  
Specialty two:  
 <answer here>