COMP 141: Data Types— Part 2

Instructions: In this exercise, we are going to study different concepts on data types.

(1) Define the function type named `int_binop` in C, that receives two integers and returns an integer. This is the type of binary operations on integers.

```
typedef int (*int_binop) (int, int);
```

(2) Now define a function named `fold_binop` that receives a function of type `int_binop`, along with an integer array and the size of the array, and return the result of folding the array, using the binary operation. The signature of the function is as follows:

```
int fold_binop (int_binop op, int a[], int size);
```

Note that folding an array using a binary operation is to repetitively apply the binary operation on the element of the array. For example, consider the following binary operation on integers:

```
int add (int x, int y) {
        return x + y;
}
```

Also consider the following array:

```
int a[5] = {1,4,2,8,1};
```

Then, `fold_binop (add, (int *) a, 5)` must return `16`.

```
//This is one way to define it:
int fold_binop (int_binop op, int a[], int size) {
   int i = 0;
   for (i = 0; i < size - 1; i++) {
      a[i + 1] = op (a[i], a[i + 1]);
   }
   return a[size - 1];
}
```

(3) Consider the following Java class definitions.

```
class A {
        public int x;
        public void setX (int x) {this.x = x;}
        public int getX () {return x;}
}

class B extends A {
        public int y;
        public void setX (int x) {this.x = x + 1;}
        public void setY (int x) {this.x = x;}
        public int getY () {return y;}
}
```

Which class is the supertype and which class is the subtype? ***A is supertype. B is subtype***

(4) In Java, subtypes can be casted to supertypes straightforwardly. This is called **upcasting**. In upcasting, the supertype variable is viewed as a reference to the subtype object. Therefore, calling shared methods would entail in subtype method invocation. Moreover, calling subtype-specific methods of that reference generates compile-time errors, since such methods do not belong to a supertype reference (at least statically).

On the other hand, **downcasting** is casting supertype to subtype. Downcasting passes static checks, but may raise runtime exception: `ClassCastException`. This exception is raised if the subtype variable refers to a supertype object at runtime.

Consider the following code excerpts for the classes defined in the previous question, and specify

- whether there is upcasting and/or downcasting in any line.
- the output (be explicit about compile-time and runtime errors)

(a)
```
A a1 = new A();
B b1 = new B();
a1.setX(5);
b1.setX(5);
System.out.println(a1.getX());
System.out.println(b1.getX());
```
no casting, output: 5 6

(b)
```
A a1 = new A();
B b1 = new B();
a1.setX(5);
b1.setX(5);
A a2 = (A) b1;
a2.setX(5);
System.out.println(a2.getX());
```
upcasting b1 to a2, output: 6

(c)
```
A a1 = new A();
B b1 = new B();
a1.setX(5);
b1.setX(5);
A a2 = (A) b1;
a2.setX(5);
System.out.println(a2.getY());
```
upcasting, compile-time error!

(d)
```
A a1 = new A();
a1.setX(5);
B b1 = (B) a1;
System.out.println(b1.getX());
```
downcasting, runtime error

(e)
```
B b1 = new B();
A a1 = (A) b1; //upcasting
a1.setX(5);
B b2 = (B) a1; //downcasting
System.out.println(b2.getX());
```
both upcasting and downcasting, output: 6

(5) Define a C program in which type `IntToInt` is defined as the type of functions that receive an `int` and return an `int` (i.e., `Int -> Int`, in Haskell syntax). Next, define a function with the following signature: `void map_array (IntToInt f, int a[], int size)`.

This function must apply function `f` on each element of array `a` and store the result in `a`.

For example, consider `int a [5] = {5, 8, 4, 7, 2}`, and function `int times2 (int x) { return x*2;}`. Calling `map_array (times2, a, 5)` must result in array `a` with the following content: 10, 16, 8, 14, 4.

Report the excerpt of your C code that includes definition for `IntToInt` and function `map_array`.

```
typedef int (*IntToInt) (int);

void map_array (IntToInt f, int a[], int size) {
        int i = 0;
        for (i = 0; i < size; i++) {
                a[i] = f (a[i]);
        }
}
```

(6) In funcional programming, functions are used to change the type of input value.

(a) What is the Haskell function that transforms the input value into a string? For example, if it receives `True`, it returns "`True`". ***show()***

(b) Use the function given in part (a) to transform the following data to string in GHCi. Report your usage of the function and the associated results.

(i) `pi`
(ii) `[1..4]`
(iii) `("hello", 'y')`
(iv) `head`

```
ghci> show(pi)
"3.141592653589793"
ghci> show([1..4])
"[1,2,3,4]"
ghci> show(("hello", 'y'))
"(\"hello\",'y')"
ghci> show(head)
Typing error! Functions are not Showable.
```

(c) What is the Haskell function that transforms the input string into a value of a given type? For example, if it receives "`True`", it returns `True`. ***read()***

(d) Use the function given in part (c) to transform the following strings to the expected structured data in GHCi. Report your usage of the function and the associated results.

(i) "`72.3`"
(ii) "`(True,'u')`"
(iii) "`[1,2,3,4]`"

```
ghci> read("72.3") :: Float
72.3
ghci> read("(True,'u')") :: (Bool, Char)
(True,'u')
ghci> read("[1,2,3,4]") :: [Int]
[1,2,3,4]
```