NAME: _____

CS 330 Final

Spring 2016

Doodle here.

On the following pages, write things that are true and relevant.

1. *Soul Your Cell*

   In Ruby, one can allocate an array and initialize each element with a block:

```
squares = Array.new(5) do |i|
  i ** 2
end
puts squares # prints [0, 1, 4, 9, 16]
```

   In this problem, you will add a method to accomplish the same task in Java 8 using lambdas and `ArrayList`.

   (a) Write an interface `Elementer` declaring the single method needed to construct an element.

   (b) Write a method `makeList` that behaves like `Array.new` in the Ruby code above. Have it return an `ArrayList<T>` that's been filled with elements generated by an `Elementer`.

(c) Call `makeList` to generate the list `[1, 3, 5, 7, 9, 11, 13, 15, 17, 19]`. Provide a lambda to generate the elements.

(d) Call `makeList` to generate the list `['A', 'B', 'C', 'D', 'E', 'F', 'G']`. Provide a lambda to generate the elements.

2. *Bahra Biskvit*

   Consider the following C++ code:

```
void print1(std::string s) {
  std::cout << &s << std::endl;
}

void print2(std::string *s) {
  std::cout << &s << std::endl;
}

void print3(std::string &s) {
  std::cout << &s << std::endl;
}

int main(int argc, char **argv) {
  std::string text = "Sağlam qida üçün, sağlam məhsul!";
  std::cout << &text << std::endl;
  print1(text);
  print2(&text);
  print3(text);
  return 0;
}
```

   Explain the output when you see when you compile and run this code. Why are the printed addresses what they are?

3. *Hide the Loops*

   (a) First, write two helper functions (complete with type signatures) in Haskell that we'll use in the second half of this problem:

      i. Write a function `bitcoinsToUSD`. It accepts a number of bitcoins and returns their worth in US dollars. As of May 13, 2016, one bitcoin is worth $458.12.

      ii. Write a function `isPowerOfTwo`. It accepts an integer number and returns true if the number is a power of two and false otherwise.

(b) Now, consider the type signatures of three canonical higher-order functions in Haskell:

```
map :: (a -> b) -> [a] -> [b]
filter :: (a -> Bool) -> [a] -> [a]
foldr :: (a -> b -> b) -> b -> [a] -> b
```

Determine which of these functions most directly suits each of the following problems, and use it to implement the function. The behavior of the functions may be accurately inferred from their example invocations.

i. `bitworths [5000, 10, 4]` → `[2290600.0,4581.2,1832.48]`
`bitworths [0, 1]` → `[0, 458.12]`

ii. `inBoth [2, 4, 6, 8] [0..4]` → `[2, 4]`
`inBoth [100, 117, 230] [99, 100, 101]` → `[100]`

iii. `reverse' [0..4]` → `[4, 3, 2, 1, 0]`
`reverse' [1000, 999, 4]` → `[4, 999, 1000]`

iv. `powers2 [1..50]` → `[1, 2, 4, 8, 16, 32]`
`powers2 [100..250]` → `[128]`

4. *Heart-to-heart*

   You are at the park with your 9-year-old child, each of you in a swing that's not swinging. The two of you stare at the dandelions in the surrounding lawn. They have shed their seeds and now stand bald and purposeless. Your child says, "Why are there so many programming languages?" How do you respond?

5. *Nounocracy*

Thanks to higher-order functions, inheritance, and polymorphism, you finished the day's work in the first half-hour. Now, to expand your mind, you are reading the latest issue of *OOPs!* (Object Oriented Puzzles), and you run across this challenge:

Uh oh! Archie `rm`'d his hierarchy! All he has left is his `main` and its output:

Code

```
int main(int argc, char **argv) {
  Child c = Child();
  Parent &p = c;
  Grandparent &g = c;

  g.print1();
  std::cout << "---" << std::endl;
  g.print2();
  std::cout << "---" << std::endl;

  p.print1();
  std::cout << "---" << std::endl;
  p.print2();

  std::cout << "---" << std::endl;
  c.print1();
  std::cout << "---" << std::endl;
  c.print2();

  return 0;
}
```

Output

```
p2
c2
g1
---
p2
c2
---
p2
c2
g1
p1
---
p2
c2
---
p2
c2
c1
---
p2
c2
```

Archie remembers that each method in the hierarchy printed out the first letter of its class and the number of its `print` method. For example, `Grandparent::print1` printed `g1`. The methods may have done other stuff too—Archie can't remember! Help Archie rewrite his classes `Grandparent`, `Parent`, and `Child`.