Due Friday, July 17th at 11:59pm (with 6 hour grace period).

No late assignments will be accepted!

Introduction

This homework has two parts: in Part A you will write up answers to analytical problems related to the lectures in the past week, both to confirm and extend your understanding of the abstract principles discussed; in Part B you will write code to implement this understanding, and to practice your Java coding skills. I suggest you read this whole assignment carefully and for Part B, it is definitely worth thinking about your solution for a bit before launching your IDE and beginning to type.

Consult the lecture slides and the Java resources linked on the class web page needed.

Part A: Analytical Problems (10 points)

The questions below are designed to reinforce the material exposed to you at lecture and from assigned readings. You MUST do these by hand (without typing them into IntelliJ or Eclipse) to verify understanding; you may of course check your answers, but if you do not do these by hand first, you will be unable to answer such questions on a test!

NOTE: Please submit your answers to the Part A problems as text files (.txt) and do NOT submit .pdf, .jpg, or any other kind of file.

1. What’s wrong with the following program? State as many errors as possible that (individually) would prevent the program from compiling without error:

```java
public class test1 { 
    public static integer f(x) { 
        if(x = 5) 
            x = 1; 
    } 
    public static void main [String[] argv] { 
        System.println('Hello World!') 
    } 
}
```

2. What would the following program print out? You MUST be able to do this kind of problem WITHOUT simply running the code and pasting the result into your answer.

```java
public class test2 { 
    public static void main (String[] argv) 
    { 
        System.out.println( (double) (int) 3.1415 ); 
        int i = 10; 
        while (i >= 1) 
            System.out.println(i); 
            --i; 
    } 
}
```

3. What would the following program print out? You MUST be able to do this kind of problem WITHOUT simply running the code and pasting the result into your answer.

```java
public class test3 { 
    public static void main (String[] argv) 
    { 
        int[] A = { 1, 1, 1, 1 }; // this initializes an array of length 4 with all values being 1 
        for(int i = 0; i < 4; ++i) { 
            for(int j = i-1; j >=0; --j) { 
                A[i] += A[j]; 
            } 
        } 
        for(int i = 0; i < 4; ++i) 
            System.out.println(A[i]); 
    } 
}
```

4. What would the following program print out? You MUST be able to do this kind of problem WITHOUT simply running the code and pasting the result into your answer.

```java
public class test4 { 
    public static void main (String[] argv) 
    { 
        int[] A = { 1, 2, 3, 4, 5, 6 }; 
        for(int i = 0; i < A.length; ++i) { 
            if(A[i] % 2 == 0) 
                continue; 
            else if(A[i] > 4) 
                break; 
            else 
                A[i] *= 2; 
        } 
        for(int i = 0; i < A.length; ++i) 
            System.out.println(A[i]); 
    } 
}
```

5. In the following program, for each name A, B, C, D, E, F, G of a variable or method, state the locations (1 to 9) which are in the scope of the variable. For example, for location A you would write "1 - 9" or "all".

```java
public class ScopeTest { 
    public static int A = 2; 
    // location 1 
    public static void B( int C ) { 
        // location 2 
    } 
    public static void main (String[] argv) 
    { 
        // location 3 
    } 
}
```

// location 3
int D = 2;
// location 4
for( int E = 0; E < 10; ++E ) {
  // location 5
  if( D < A ) {
    int F = 1;
    // location 6
  } // location 7
} // location 8
// location 9
public static int G = 2;
}

I REPEAT: I strongly recommend you work out the problems WITHOUT trying them in Java, and only after thinking them through and coming up with an answer then try them out in Java. If you simply type them in to get the answer without much thinking, then you will be unable to solve such problems on a test, and MOST IMPORTANTLY, you won’t learn the principles being taught!

Write your answers to these questions as a file hw02.txt, and hand in as part of your homework submission in Gradescope.

Part B: Programming Problems (90 pts)

Useful String Methods (Please read this through before starting the first problem)

In this homework, we will continue to explore using Java to do input and output, and in particular focus on using Strings; you should read through the follow material before starting the problems. In addition, here is a simple reference on Strings. Googling "Java String library" or "Java Character library" will produce many references with explanations about these libraries For full details, he canonical reference on these libraries is the Oracle online Java page, which contains every detail you need to know; String, Character. For this assignment, these are a bit of overkill, but you should know about them.

Finally, if you don’t mind the fact that he uses Dr. Java instead of IntelliJ or Eclipse, it is worth looking at Aaron Steven's Java videos 14 and 14b (Aaron is one of our BU instructors).

Note: Each method is followed by examples, assuming the initial declaration String str = "Hi There" and where "=>" means "returns the value": these are methods that are called in conjunction with the String variable, using the "dot notation" as shown in the examples.

String str = "Hi There";

charAt( n ) -- returns the character at the given index n in the string (starts at 0):
   str.charAt(1) => 'i'

NOTE: You CAN NOT change a String by doing for example:
   str.charAt(1) = 'h'

Strings are immutable, and to change a String you have to create a new one!

toLowerCase() -- converts all letters to lower case;
   str.toLowerCase() => "hi there"

equals( str2 ) -- Returns true or false depending on whether the string is equal to the parameter
   str.equals("Hi There") => false
   str.equals("Hi There") => true

Note: DO NOT use == to compare Strings, this is almost always the wrong thing to do!

replace(...) -- Create a new String where all occurrences of one substring are replaced by another -- the original String is unchanged.
   str.replace("Hi", "Hello") => "Hello There"
   str.replace("e", ",") => "Hi Thr"
   str = str.replace("e", ","); // remove all occurrences of the character 'e' from str

replaceFirst(...) -- Create a new String where the first occurrence one substring is replaced by another -- the original String is unchanged.
   str.replaceFirst("Hi", "Hello") => "Hello There"
   str.replaceFirst("e", ",") => "Hi Thr"
   str = str.replaceFirst("e", ","); // remove the first occurrence of the character 'e' from str

length() -- Return the length (number of chars in) the String. NOTE that this is different from the way you find the length of an array: A.length
   str.length() => 8

split("\s+") -- Separate each String into separate strings using the white space between them as a separator and put them in an array (here we are giving you the parameter which does this, others can be used as well).
   String[] words = str.split("\s+");
   // words is now the array ["Hi", "There"]

Useful Character library methods (the first three of these return boolean values; read about here)
Problem B.1 (30 points)

Now download `StringTest.java` and run it; you will see that it inputs a sequence of strings and simply prints them out between double quotes. It will continue to do this until you type Control-d (hold down the Control key and hit the letter 'd').

For this problem, you must rewrite `StringTest.java` into a program `PalindromeTest.java`, which will read in a string, which is assumed to be an English sentence, and check whether it is a palindrome (reads the same backwards as forwards). Here are the steps you should follow:

1. Read in the string and convert it to all lower case using `toLowerCase();`
2. Remove any character which is neither a letter nor a digit; you can use `Character.isDigit(...)` and `Character.isLetter(...)` to test for this. All punctuation and white space will be remove. (Hint: use `replace(...)` to replace any non-letter non-digit by the empty String """).
3. Using `charAt(...)` check if the word is a palindrome by checking if each letter is the same as the letter in its "mirror image" position; for example "Taco Cat" and "Toot!" are palindromes:

```
tacocat
  toot
```

but "honor" is not.

NOTE: You may NOT do the problem by converting the String to a char array first; I want you to use the String methods! Another possible way to do this problem is to reverse the String and then compare if they are equal; again, you may NOT do the problem this way, but must follow my requirements above.

Test your code on the following palindromes:

```
A man, a plan, a canal, Panama!
Go hang a salami, I'm a lasagna hog!
Campus Motto: Bottoms up, Mac!
7/1/17 // A recent palindrome date!
Are we not pure "Ho siri!" Panama's moody Noriega brags. It is garbage! Irony dooms a man; a prisoner up to new era.
(Other examples are given at the bottom of this lab!)
(Warning: This last one might cause a problem because there are various kinds of "smart quotes" which are different from the simple ASCII double quotes; if all the others work and this one doesn't, don't worry about it!)
```

Also test your code on a non-palindrome example -- such as the word "palindrome" itself, or this sentence!

Just print out "Palindrome!" or "Not a palindrome!" depending on your answer. Here is an example of what your program should do:
Problem B.2 (30 points) : Big Int Library: Creating your own static library class

In this problem you are going to practice creating a static library (like Math) to solve the problem that in Java the int type can only represent integers in the range [-2,147,483,648 ... 2,147,483,647]; this is not big enough for storing, for example, the population of Earth (7.125 billion). The Java libraries contain a class BigInteger to solve this problem, but we are going to write our own library, BigInt, with only some of the functionality of the Java library. Note that this library, like Math, is a static container for methods; all the members of the class will be static.

Representation of Big Integers by Arrays of Digits

You must complete the template code BigInt.java to implement your solution.

We will represent our big integers by large arrays (we will use arrays of size 20, but they could be larger) of integers representing individual digits:

```
public static final SIZE = 20;                        // this gives the length of the arrays, and will occur in BigInt as a constant
```
public static int[] A[] = new int[BigInt.SIZE]; // this is an example of an array -- it will not occur in the class BigInt

An integer such as 57431 is represented by an array A containing integers which are in the range [0 -- 9] (i.e., digits) with as many leading zeros as necessary to fill up the entire array:

00000000000000057431

would be presented as

```
0  1   2  14  15  16  17  18  19
+-----------           ------------------------+
A:  | 0 | 0 | 0  ......   | 0 | 5 | 7 | 4 | 3 | 1 |
+-----------           ------------------------+
```

The ints in the array must all be single digits. Note that the BigInt for 0 will have SIZE-1 leading 0's and the last digit will be a significant digit, not a leading 0. We will not represent negative integers. We will also use a "Error Big Int" called NaBI (Not a Big Int) consisting of a -1 in slot 0, with the remaining values set to 0:

```
0   1   2  14  15  16  17  18  19
+-----------           ------------------------+
NaBI:  |-1 | 0 | 0  ......   | 0 | 0 | 0 | 0 | 0 | 0 |
+-----------           ------------------------+
```

The Methods of the BigInt Static Library

The library will contain the following methods, which assume that the integer arrays are of length SIZE.

```
public static int[] intToBigInt(int n) -- Convert the integer n into a big integer; if n < 0, then return NaBI; note that n won’t overflow the array, since it’s an int.

public static int[] stringToBigInt(String s) -- Convert the String s into a big integer; if s does not represent a legal big int (i.e., contains a character other than '0' .. '9' or is longer than SIZE characters) return NaBI.

public static String bigIntToString(int[] A) -- Return a String representation of the big integer (with leading zeros suppressed); if any member of A is NOT a digit (e.g., is negative or > 9) return "NaBI".

public static int compareTo(int[] A, int[] B) -- Compare the two big integers A and B and return -1, 0, or 1, depending on whether A < B, A == B, or A > B, respectively.

public static int[] add(int[] A, int[] B) -- Add two big integers and return a new array representing their sum; if the result overflows, i.e., contains more than SIZE digits, return NaBI.
```

Algorithms

The algorithms for manipulating these big integers are simply adaptations of the rules for addition and comparison of integers. To compare two big ints, simply go through the two arrays from the left to right, and when you get to two digits which are different, return -1 (if the digit in A is smaller) or 1 (if the digit in A is bigger); if you get to the end of the arrays and all the digits are the same, return 0.

To add two arrays of digits, we must go from right to left through the arrays, adding the digits and keeping track of the carry at each point:

```
carry:  1   0   1   1
...   0 | 5 | 7 | 3 | 3 | 9 |
+-----------           ------------------------+
equals  57339

...   0 | 0 | 4 | 5 | 9 | 8 |
+-----------           ------------------------+
equals  4598

sum:   ...   0 | 6 | 1 | 9 | 3 | 7 |
+-----------           ------------------------+
equals  61937
```

The addition will overflow (create a number larger than 20 digits) if the sum of the left-most two digits is larger than 10, in which case you should return NaBI.

Completing your Solution

You must fill out the "stubs" in the template BigInt.java. Note that this template will compile, because I have added dummy return statements just for that purpose; obviously you will need to change these when you add your code.

You must follow these requirements for this problem:

- You should examine the method stubs and fill them in with appropriate code to implement the functionality described above.
- Use my descriptions above (edited or expanded as you wish) as comments before each of the methods; any lines that seem to require more explanation (e.g., to yourself trying to understand what you were thinking, 2 years from now at 4am).
- You may add tests to the main method as you develop your code, but remove them before submission.
- Remove any debugging code before submission.

Problem B.3 (30 points): Set Data Type: Creating a dynamic Abstract Data Type

In this problem, we are going to to create an Object to represent a set of integers. In mathematics and computer science, a set is a collection in which order does not matter and there are no duplicates. You will write a program named Set.java which implements sets using arrays. Each instance of the Set class will be an object representing a set. The only member of the class which will be static is the main method, which contains the testing code.

The template for this class is here: Set.java.

We will represent our sets by arrays of arbitrary size (but starting with size 10) with a pointer next indicating the next available slot in the array after all the elements of the set:

```
private final int SIZE = 10;  // initial length of the array, may be resized
private int[] S;              // array holding the set
private int next;             // pointer to next available slot in array
```

...
Thus, a set \{2, 3, 7, -3\} would be represented as follows:

\[
\begin{array}{cccccccccc}
0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 \\
\hline
2 & 3 & 7 & -3 & 0 & 0 & 0 & 0 & 0 & 0
\end{array}
\]

and the empty set [] would be represented as:

\[
\begin{array}{cccccccccc}
0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 \\
\hline
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{array}
\]

The idea is that you can represent a set of up to SIZE integers, and they are stored, without duplicates, in the indices from 0 to next-1. The values from next to S.length-1 do not matter—they do not need to be 0s. When manipulating the array using a for loop, you should use next as the for loop bound, NOT S.length.

The Methods of the Set Object

You must write the following methods, which are provided as stubs in the template linked above:

```java
public Set() --
    Default constructor; constructs this set as an instance of the empty set.

public Set(int[] A) --
    Construct this set consisting of exactly the elements of A (which, you may assume, does not have duplicates); A can be
    of arbitrary length (it may not be smaller than SIZE). (Hint: create an empty set and use insert(...) to add the elements,
    which may trigger a resize of the array.)

public Set clone() --
    Return an exact copy of this set (hint: use the previous constructor).

public String toString() --
    Return a string representation of this set, e.g., \{2,3,7,-3\} or []; you may not use Array.toString(...) to do this; you may
    create a char array from the array S and use the String constructor (which will accept a char array as initializer), or you
    may build up the String one character at a time using concatenation.

public int size() --
    Return the number of elements in this set.

public boolean isEmpty() --
    Return true if this is the empty set (has no members) and false otherwise.

public boolean member(int n) --
    Return true if n is in the set and false otherwise.

public boolean subset(Set T) --
    Returns true if this set is a subset of T, that is, every member of this set is a member of T, and false otherwise.
    (Hint: use member.)

public boolean equal(Set T) --
    Returns true if this set and T have exactly the same members. (Hint: use subset.)

public void insert(int k) --
    If the integer k is a member of the set already, do nothing, otherwise, add to the set; if this
    would cause an ArrayOutOfBoundsException, call resize() (provided in template code) to increase size of array
    before doing insertion.

public void delete(int k) --
    If the integer k is not a member, do nothing; else remove it from the set; you must shift all elements
    which occur after k in the array to the left by one slot.

public Set union(Set T) --
    Return a new set consisting of all of the elements of this set and T combined (without duplicates).
    (Hint: make a clone of this set, and insert all elements of T into the clone.)

public Set intersection(Set T) --
    Return the intersection of this set and T. (Hint: make a new empty set, and for every element of this
    set, if it also occurs in T, insert it into the new set.)

public Set setdifference(Set T) --
    Return the setdifference of this set and T. (Hint: make a new empty set, and for every element of this
    set, if it does NOT occur in T, insert it into the new set.)
```

Important Note: The last three methods should make new sets, and NOT modify their input sets in any way! Insert and delete will modify the set. Look for opportunities (suggested in the hints) to reuse your own code by implementing methods by using other, simpler methods. For example, equals can be implemented simply (one line of code) using two calls to subset, and setdifference is VERY similar to intersection!

Note that there is no requirement for error checking in this assignment.

Note that we will be using array resizing to handle the problem of overflow -- I have provided a method resize() in the template and specified when it should be used (in insert); we will discuss this in lecture soon, but you do not need to know the theory to simply use the method.

Completing your Solution

You must fill out the "stubs" in the template Set.java linked above. Note that this template will compile, because I have added dummy return statements just for that purpose; obviously you will need to change these when you add your code.

You must follow these requirements for this problem:

- You should examine the method stubs and fill them in with appropriate code to implement the functionality described above.
Use my descriptions above (edited to remove my comments on how to implement!) as comments before each of the methods; any lines seem to require more explanation (e.g., to yourself trying to understand what you were thinking, 2 years from now at 4am).
You may add tests to the main method as you develop your code, but remove them before submission.
Remove any debugging code before submission.

Submission

Submit all files into Gradescope by the due date and time; the time of submission will be when the LAST file is uploaded.

Here is a check list to follow in order to make sure you have completed this homework correctly:

Submit files:
- hw02.txt
- PalindromeTest.java
- BigInt.java
- Set.java

Check:
- You have formatted your code so it is neat and readable (by a human), and written appropriate comments (including a "header comment" at the top of the file);
- You have removed all comments inherited from my template which do not relate to your solution (e.g., '// return error, just to get it to compile')
- You have removed any testing code which was added during programming and testing, and are no longer relevant to the program.
- You have submitted files named EXACTLY as specified above (including capital letters).