CS 514. Midterm 2. 80 points total.
Your name: Qianyun Wang

Please insert your answers into this document, using a different font or color from mine so that I can find your answers easily. If you would prefer to do the tracing problem or class diagram by hand or in another tool, you may do so and attach a photo, screenshot or PDF to the end of this document.

Exam rules:

- The exam must be submitted on Canvas by Monday, Nov 9 before the start of class (10:30am).
- The exam is "open book, open note." Specifically, here's what that means:
    - You may refer to the textbook, the Java documentation, or any of the course materials on Canvas when answering questions. If you are using this for reference or background, there's no need to provide citations. For example, if you used the textbook or slides to remember how an array works, you do not need to cite that.
    - If you are **directly quoting** either the text or the course materials, you must provide a reference. For example, if you were answering a question about inheritance and wanted to use the Horstmann text's definition, you would say: inheritance is "the relationship between a more general class and a more specialized class." (Horstmann, Ch 9). If you are writing this in your own words, there is no need to provide a reference.
    - If you are using information from some other source (not necessarily encouraged), you should provide a reference indicating where you got this from. For example, if you were discussing inheritance and found through Wikipedia that C++ supports multiple inheritance but Java does not, and wanted to add this to an answer, you should say something like: unlike Java, C++ supports multiple inheritance. [per Wikipedia]. Please keep in mind that just because something is on the Web does not make it true; you are responsible for the correctness of everything in your exam.
- Each student is expected to complete their own exam. Working together or asking others to assist or complete portions of the exam for you is expressly forbidden.
- All students are expected to abide by the USF Honor Code.
  https://myusf.usfca.edu/academic-integrity/honor-code

  By submitting this exam on Canvas, you confirm that you are abiding by this Honor Code, and understand the consequences of violating this code.

1. You've just finished implementing Project 2. The following set of questions ask you to consider possible extensions and how you would implement them. Please be specific and precise in your answers.

   (5 points) Suppose that we decided to add an *exponentiation* operator, denoted ^.
   For example, 4 ^ 3 = 64. We can also use it in larger expressions.
   For example, 2 ^ 3 + 5 ^ 2 = 32.

   How would you modify the lexer and parser to account for this?

   Lexer: Adds support to interpret *exponentiation* operator in getOperator (buffer.charAt(ref) == '^')

   Parser:
   1. **Set highest priority for *exponentiation* operator (precedence.put("^", 3));**
   2. **Adds supports for exponentiation operator in *public double eval(SymbolTable table)* function. To be specific: i*f (val.equals(("^"))) { return Math.pow(left, right); }***

   (10 points) Suppose that we decided to add support for *lists*. In other words, we would like to be able to do:
   >> x = [1, 2, 4]
   >> x
   [1,2,4]

   - How would you extend the Lexer to handle this?
   - How would you extend the Parser to handle this?
   - How would you store the list referred to by x in the symbol table?

   Assumes all the elements inside the list are the same type (int or float).
   Lexer: Add Another type called "LIST" inside Lexer class and a method called getList, which will return the token type = "LIST" and value = "[1,2,4]" for example
   Parser: In parseAssignment() function, Node pe = parseExpression(tokenList.subList(2, tokenList.size())). Then we check

```
if (pe.type != "Lexer.LIST") {
        double val = ep.eval(table);
        table.symTable.put(id.val, val);
} else {
```

```
                List<Double> l = parseList(ep.val); // which takes a string "[1, 2, 4]" as input,
and output a arrayList of Double [1, 2,  4];
                table.symTableList.put(id.val, l);
        }
```
In evaluate() function, change the return type from double to String.

```
        public String evaluate(SymbolTable table) {
                // first check if it's LIST type;
                if (table.symTableList.containsKey(root.val)) {
                        List<Double> list =  table.symTableList.get(root.val);
                        String listString = String.join(", ", list); // convert to string
                        return listString;
                } else {
                        return String.valueOf(root.eval(table));
                }
        }
```
Symbol table: Add another variable Map<String, List<Double>> symTableList to store the list object.

(5 points) Suppose that we then want to add in the ability to index into a list. E.g.:

```
>> x = [1, 2, 4]
>> x[1]
2
```

- How would you modify the Lexer to handle this?
- How would you extend the parser to handle this?
- How would you extend eval() to handle this?

Lexer: I plan to treat "x[1]" as an identifier. So I will relax the definition of Identifier. In function getIdentifier(), if the character is '[' or ']', it's still considered as valid.

Parser: there is no change in the parser

Eval: I will add one step to parse the new defined identifier.

```
        if (table.symTable.containsKey(val))
        {
                return table.symTable.get(val);
        } else if (table.functionTable.containsKey(val)) {
                return table.functionTable.get(val).evaluate(table);
         }
        //  Then we will check if val is in the format of "x[1]", if so, we will parse it.
        if (isValidIndexFormat(val)) {
                String newId = parseId(val); // which will output x
                int idx = parseIdx(val); // which will output 1;
                if (table.symTableList.contrainsKey(newId)) {
                    return table.symTableList.get(newId).get(idx);
                }
        }
        throw new IllegalArgumentException(" identifier not in symbol table");
```

(5 points) Suppose that we are building an application that needs to keep track of student records. It has the following characteristics:

- 1) We will be repeatedly adding and deleting records.
- 2) We need to be able to print the records in alphabetical order.
- 3) We want to be able to quickly locate a specific student record.

What data structure would you recommend we use to store these records? Please explain your answer.

I will use TreeMap to store these records. The key will be student identification, and the value will be the full records. It's because the Operation 1) (Adding and deleting number) and Operation 3) (Locate a specific record) is relatively fast with O(logn) complexity. Also all the keys in the TreeMap are already sorted, so Operation2) to print out the records in alphabetical order is really fast with O(n) time complexity. Also the space complexity is O(n)

(5 points) Suppose that our problem changes.
We know in advance that there are 2000 student records. We need to be able to add and access individual records as quickly as possible, but we no longer care about being able to print them in alphabetical order.

What data structure would you now recommend we use? Please explain your answer.

I will use HashMap to store these records. The key will be student identification, and the value will be the full records. The keys in the HashMap are not sorted, which is fine in this case as we don't need to print them in alphabetical order. But Add and Access Operations are super fast with O(1) complexity compared to O(logn) for TreeMap.

(5 points) Suppose that we are building an application that will let us navigate through places in San Francisco. We break the city map into a 100x100 grid, and for each cell we want to store a string that will indicate the contents of that cell. (e.g. “USF”, or “City Hall”). We want to be able to:

- Print the contents for a particular (x,y) location
- Update the contents for an (x,y) location.
- For any x,y location, quickly find the neighboring cells and their contents.

What data structure would you recommend that we use? Please explain your answer.

I will use 2D String arrays data structure like String[][] grid = new String[100][100]; where each cell stores the corresponding string.

The major reasons are:
1) Print the contents for a particular (x,y) location is O(1) time complexity using grid[x][y]
2) To update the contents for an (x,y) location is O(1) time complexity using grid[x][y] = newString.
3) For any x,y location, to find the neighboring cells and their contents is also O(1) time complexity using (grid[x-1][y], rid[x+1][y], grid[x][y -1] and grid[x][y+1])

(10 points) Consider the following code for working with linked lists. Find each bug, explain why it is a bug, and correct it. (there are at least 5).

```
public class LinkedList<Type> {

     protected Node<Type> head;

     class Node<Type> {
        Type data;
        Node<Type> next;

         Node(Type d) {
            this.data = d;
            this.next = null;
        }

        Node() {
           this.data = null;
           this.next = null;

    }

    public LinkedList() {
        head = null;
    }

    public void addInFront(Type item) {
        Node n = new Node(item); // We need to provide item to construct Node
n)
        ~~Head = n;~~
        n.next = head;
        head = n;  // We need to first connect head to n's next element and
then replace head with n.
      }

    public void remove (Type item) {
     Node temp = head;
     for (int i = 0; i < this.length(); i++) { // there is no length() method,
need to use while (temp.next != null) instead
          if (temp == item) {                 // 1) cannot compare two
different types of objects between Node temp and Type item. 2) Also we need to
compare between temp.next and item
               temp.next = temp.next.next;
          } // we need to deal with the case when it's not equal. Then need to
set temp = temp.next;
     }
      // here is the correct implementation
      Node dummy = new Node();
      dummy.next = head;
      Node temp = dummy;
      while (temp.next != null) {
            if (temp.next.data.equals(item)) {
                  temp.next = temp.next.next;
            }
            else {
                  temp = temp.next;
            }
      }
      head = dummy.next;
    }
}
```

(15 points) For each of the following code snippets, modify them, using streams, to produce the effect indicated in the comment.

```
// change this to only count words with more than 4 letters.

List<String> stringList = new LinkedList<>();
stringList.add("cat");
stringList.add("dog");
stringList.add("bunny");
```
`long count = stringList.stream().filter(w -> w.length() > 4).count();`

```
// change this to produce a list of words beginning with 'd', with all letters
upper case.

List<String> stringList = new LinkedList<>();
stringList.add("cat");
stringList.add("dog");
stringList.add("bunny");
```
`List<String> results  = stringList.stream().filter(w ->`
`w.startsWith("d")).map(w -> w.toUpperCase()).collect(Collectors.toList());`

```
// change this to generate a new collection of words with length > 3,
// with duplicates removed.

List<String> stringList = new LinkedList<>();
stringList.add("cat");
stringList.add("bunny");
stringList.add("dog");
stringList.add("bunny");
```
`List<String> results = stringList.stream().filter(w -> w.length() >`
`3).distinct().collect(Collectors.toList());`

(3 points) What is the purpose of the Comparable interface? Give an example of where it would be useful. Why is it better to make Comparable an interface rather than an abstract class that we subclass?

Classes which implement Comparable interface will be able to be sorted with a sorted method called. For example, in the LinkedList project, we implemented Comparable interface in Node class and implemented the compareTo Method. Then later on, we can use prev1.compareTo(prev2) to compare two Nodes to implement the sortedLinkedList class. Or sort() method can automatically work with Node class.

The semantics are different between interface and abstract class. To inherit from a parent class indicates a isA relationship. But to implement an interface indicates different classes share the same API or functionality. Here, we want to define the abstract method shared by different classes

(3 points) What is the purpose of generics? How do generics allow for code re-use?

B**y using generics, we can define data structure or method without caring about the type.And we can use whatever data type we care about to initialize it. This way, we only need to write code for the generic type once, and can use it for whatever data types instead of writing one for each specific type.**

(3 points) What is the purpose of the finally block?

It can help to do some final clean up like closing files or connections to exit gracefully. It contains the crucial statements that will always be executed no matter whether the exception happens or not.

(3 points) Why is the following code bad practice?

```
try {
    Scanner input = new Scanner(new File("myinput.txt"));
    String[] buffer = input.readLines();
} catch (Exception e) { }
```

1. **It's an empty catch block which doesn't handle the exception at all and it's a source of lots of bugs.**
2. **It uses expectation instead of one of the subclasses like FileNotFoundException and IOException, from which we can't determine exactly what happens.**

(8 points) In one of our reflections, you learned about *open source* software.

What does it mean for software to be open source?

The original source code of the software is freely available and anyone can inspect, modify and enhance it for reusability and accessibility. It's the kind of software that can be redistributed and modified based on the requirements of the user.

Why might a developer choose to release their project as open source?

1. **The best developers want to share their ideas. And it allows experts from all over the world to consult your project. [Refer: https://www.shopsys.com/6-reasons-why-we-moved-from-closed-to-open-source-and-you-should-too-9d7007cd71f8/]**
2. **You may not have a grand strategy or any intention of shaping the industry with your project. Releasing a project as open source allows others to adapt and build on top of your project. When people build on top of your project, they are invested in your success as well as their own. Publishing your project under an open source license can encourage adoption of a standard. [Refer: https://opensource.google/docs/why/]**

Refer:
https://www.shopsys.com/6-reasons-why-we-moved-from-closed-to-open-source-and-you-should-too-9d7007cd71f8/
https://opensource.google/docs/why/

Recall our reflection about Creative Commons. How does Creative Commons provide additional flexibility for content creators beyond traditional commercial licenses?

Creative Commons provides free, easy-to-use copyright licenses that provide a simple, standardized way to give the public permission to share and use your creative work. CC licenses let you easily change your copyright terms from the default of "all rights reserved" to "some rights reserved." Creative Commons licenses are not an alternative to copyright. They work alongside copyright and enable you to modify your copyright terms to best suit your needs." [Refer: https://alj.artrepreneur.com/use-creative-commons-license/]

**Convenience: under traditional copyright, if you want others to use your work, you must grant licenses to each person individually, whereas Creative Commons empowers you to grant permission to everyone automatically. [Refer: https://opensource.com/article/20/1/what-creative-commons
]**