# Sample Midterm

CS136
Williams College

This is a *closed book* exam. You have one hour and 15 minutes to complete the exam. All intended answers will fit in the space provided. You may use the back of the preceding page for additional space if necessary, but be sure to mark you answers clearly.

Be sure to give yourself enough time to answer each question— the points should help you manage your time.

In some cases, there may be a variety of implementation choices. The most credit will be given to the most elegant and efficient solutions.

| Problem | Points | Description | Score |
|---|---|---|---|
| 1 | 14 | True/False | |
| 2 | 10 | Static | |
| 3 | 26 | Creating a Set class | |
| 4 | 15 | Recursion on Lists | |
| 5 | 12 | Big-O | |
| Total | 77 | | |

I have neither given nor received aid on this examination.

Signature: _____

Name: _____

## 1. (*14 points*) ..... True/False

1. True/false statements (2 points each). Justify each answer with a sentence or two.

a. Two instances of `class Association` in the `structure` package are equal if and only if their `key`s are equal, regardless of their `value`s.

b. An instance variable declared as `protected` can be accessed by any method of the class in which it is declared.

c. A binary search can locate a value in a sorted `Vector` in $O(\log n)$ time.

d. A binary search can locate a value in a sorted `SinglyLinkedList` in $O(\log n)$ time.

e. If a method that has no preconditions is called, all of that method's postconditions should be guaranteed to be true when the method returns.

f. The Unix command `cp /path/to/directory` changes your current working directory to `/path/to/directory`.

g. Instance variables are specified in an interface file.

## **2**. (*10 points*) ..... Static

Consider the following Java program:

```
class Container {
    protected int count;
    protected static int staticCount;

    public Container(int initial) {
        count = initial;
        staticCount = initial;
    }

    public void setValue(int value) {
        count = value;
        staticCount = value;
    }

    public int getCount() {
        return count;
    }

    public int getStaticCount() {
        return staticCount;
    }
}

class WhatsStatic {

    public static void main(String[] args) {
        Container c1 = new Container(17);
        System.out.println("c1 count=" +c1.getCount()+
                           ", staticCount=" + c1.getStaticCount());

        Container c2 = new Container(23);
        System.out.println("c1 count=" +c1.getCount()+
                           ", staticCount=" + c1.getStaticCount());
        System.out.println("c2 count=" +c2.getCount()+
                           ", staticCount=" + c2.getStaticCount());

        c1.setValue(99);
        System.out.println("c1 count=" +c1.getCount()+
                           ", staticCount=" + c1.getStaticCount());
        System.out.println("c2 count=" +c2.getCount()+
                           ", staticCount=" + c2.getStaticCount());

        c2.setValue(77);
        System.out.println("c1 count=" +c1.getCount()+
                           ", staticCount=" + c1.getStaticCount());
        System.out.println("c2 count=" +c2.getCount()+
                           ", staticCount=" + c2.getStaticCount());
    }
}
```

Answer the following questions (next page) about this code.

a. What will the output be when the program is run (`java WhatsStatic`)? Assume no exceptions occur. (4 points)

b. What memory is allocated for `Container`s `c1` and `c2` at the time the line `c1.setValue(99)` is executed? Show any existing local variables and instance variables. (6 points)

## **3**. (*26 points*) ..... Creating a Set class

In this problem you are to design a Java interface and class for a data structure which represents sets of Strings. As usual for sets, no repeated elements are allowed. Thus, the collection `"Propser"`, `"Anya"`, `"Lisa"`, `"Karl"`, `"Isabella"` is a legal set, but `"Bill"`, `"Duane"`, `"Bill"` is not. This data structure will have two methods:

- `void insert(String myString)` adds `myString` to the set if it is not already in the set; if `myString` is already in the set then `insert` does nothing.
- `boolean contains(String myString)` returns a boolean value indicating if `myString` is an element of the set.

a. Write a legal Java `interface` called `StringSetInterface` for this data structure. Include javadoc-style comments for preconditions and postconditions for the methods, as well as their return values where applicable. (You do not need to create javadoc-style comments for the parameters.) (6 points)

Name: _____

b. Suppose we decide to implement `StringSetInterface` by a class in which a `structure5 SinglyLinkedList` holds all of the elements. Write the definition of this class. This should be a full and legal Java class definition *with all method bodies filled in*. Don't forget to declare instance variables, include a constructor, and use qualifiers such as `public` and `protected` when appropriate. You need not repeat your javadoc-style comments for each method from part a. Please call your class `StringSet`. (10 points)

Name: _____

c. If `StringSet` is implemented as in part b, what would the worst-case time complexity be for the insert operation when the set has $n$ elements? (Use "Big O" notation.) (4 points)

d. Suppose we design an alternative implementation for this problem in which the set is represented by a `Vector<String>` called `strVec`. What is the worse-case complexity of insert with this representation? (6 points)

## 4. (*15 points*) ..... Recursion on Lists

(15 points) Consider the following class, `ReversibleList`, that extends the `SinglyLinkedList` class by adding a method for reversing the list.

```
public class ReversibleList<E> extends SinglyLinkedList<E> {

  public ReversibleList() {
    super();
  }

  /**
  * @post: list is reversed
  */
  public void reverse() {
    if (head != null)
      head = recReverse(head);
  }

  /**
  * @pre: current is not null.
  * @post: list headed by current is reversed;
  *        and first Node in that list is returned.
  */
  private Node<E> recReverse(Node<E> current) {
    if (current.next() == null) {  // Single-node list
      return current;
    } else {
      Node<E> newHead = recReverse(current.next());  // Explain
      // current.next() now points to final node in reversed list!
      current.next().setNext(current);  // Explain
      current.setNext(null); // Explain
      return newHead;
    }
  }
}
```

a. What is the running time of `reverse()` (3 points)?

b. Prove using mathematical induction that your answer to part a is correct. (12 points)

**5**. (*12 points*) ..... Big-O

5. Growth of functions. Using "Big O" notation, give the rate of growth for each of these functions. Briefly justify your answers (you do not need to use the definition of Big O; just explain how you found your solution). (3 points each, 12 total)

a. $f(n) = n^2 + 17n + 2001$

b. $f(n) = 3n + 5 \log_2 n$

c. $f(n) = 7n$ when $x$ is odd, $f(n) = \frac{n}{7}$ when $x$ is even.

d. $f(n) = 5n^3$ for $n < 23$, $f(n) = 37$ otherwise.