Midterm 2: Compsci 201
Form C

Prof. Astrachan

October 30, 2024

Name: _____

netid: _____

In submitting this test, I affirm that I have followed the Duke Community Standard.

Community standard acknowledgement (signature) _____

The definition of `ListNode` unless another is specified. This is the same definition as used in linked-list APT problems.

```
public class ListNode {
    int info;
    ListNode next;
    ListNode(int x) {
        info = x;
    }
    ListNode(int x, ListNode node) {
        info = x;
        next = node;
    }
}
```

This Exam is Form C, please mark your answer sheet accordingly

The method `create` below is intended for the call `create(10)` to return a list `1->2->...->9->10`, but instead the call returns a list with a single node containing zero. In general, the call `create(N)` is intended to return a linked list of N nodes in order `1->2->....->N`.

```
25    public ListNode create(int n){
26        ListNode list = null;
27        while (n >= 1){
28            list = new ListNode(0,null);
29            n -= 1;
30        }
31        return list;
```

PROBLEM 1:

As written, what is the complexity of the call `create(N)`?

A. $O(1)$

B. $O(N)$

C. $O(N^2)$

PROBLEM 2:

What statement can replace the right hand side of the assignment statement on line 28 so that the code works as intended? The only modification you can make is to the right hand side of line 28.

Write the new code in the appropriate fill-in-the blank area of the back of the bubble answer sheet. **Bubble A for this question on the front of the answer sheet.**

PROBLEM 3:

An attempt to write a correct, recursive version to solve this problem yields the code below, it does **not work as intended**.

```
43    public ListNode create(int n){
44        if (n == 1) return new ListNode(1,null);
45        ListNode afterMe = create(n-1);
46        return new ListNode(n,afterMe);
47    }
```

Which of the following is the best description of the result returned by the call `create(10)`?

A. It creates 10 nodes, but returns a pointer to one node which contains 10 and whose next field is null.

B. It creates 10 nodes, but returns a pointer to one node which contains 1 and whose next field is null.

C. It creates one node, and returns a pointer to that single node which contains 10 and whose next field is null.

D. It creates 10 nodes and returns a pointer to a node containing 10, which is the first node of a linked list: `10->9->8->...->2->1->null`

The *ListSum* APT was a required APT to write a method `sum` that adds those node values in a linked list that are greater than a threshold parameter. For example, if `list = 6->10->7->8` the call `sum(list,7)` returns 10+8 == 18 and the call `sum(list,10)` returns 0 (no values in the list are greater than 10.

In the code below, method `sumIter` is all green if the name is changed to `sum`. The method `sum` is a recursive method that is all green if two statements are added.

```
1  public class ListSum {
2      public int sumIter(ListNode list, int thresh) {
3          int total = 0;
4          while (list != null){
5              if (list.info > thresh) {
6                  total += list.info;
7              }
8              list = list.next;
9          }
10         return total;
11     }
12     public int sum(ListNode list, int thresh) {
13         if (list == null) return 0;
14         int total = 0;
15         if (list.info > thresh) {
16             // assignment statement
17         }
18         return expression with recursive call;
19     }
```

PROBLEM 4:

If `sum` is all green, is there a linked list `list`, with one or more nodes all of whose node values are greater than zero, such that the value of the call `sum(list,sum(list,0))` is greater than zero?

A. Yes, there is such a list with one or more positive values

B. No, the value returned by the call is zero for any linked list of one or more positive values.

PROBLEM 5:

If `sum` is all green, and the value of `sum(list,thresh)` is $N > 0$, what is the value of `sum(list,N)`

A. 0

B. $O(N)$

C. it cannot be determined

PROBLEM 6:

Write the assignment statement for line 16, so the code is all green, in the appropriate fill-in-the blank area of the back of the bubble answer sheet. **Bubble A for this question on the front of the answer sheet.**

PROBLEM 7:

Write the expression that includes a recursive call for line 18, so the code is all green, in the appropriate fill-in-the blank area of the back of the bubble answer sheet. **Bubble A for this question on the front of the answer sheet.**

Part of the code in a working version of `HashMarkovModel` from the P3-Markov assignment is shown below.

```
3  public class HashMarkovModel extends AbstractMarkovModel {
4      private HashMap<WordGram,List<String>> myMap;
5
6      public HashMarkovModel(int order){
7          super(order);
8          myMap = new HashMap<>();
9      }
10     public HashMarkovModel(){
11         this(3);
12     }
```

PROBLEM 8:

Execution of the statement `HashMarkovModel mod = new HashMarkovModel();` results in running the code on line 11 above (because of the call of the default constructor).

Does executing line 11 and the code that line 11 then calls result in creating a `HashMap<>` object?

A. No, the value of `myMap` will be null, perhaps until `setTraining` is called.

B. Yes, `myMap` will be initialized to point to/reference a `HashMap` because of the other constructor.

PROBLEM 9:

The code below for method `getRandomNextWord` is the same code that runs in both `BaseMarkovModel` and `HashMarkovModel`

```
41 public String getRandomNextWord(WordGram wgram) {
42     List<String> follows = getFollows(wgram);
43     if (follows.size() == 0) {
44         return END_OF_TEXT;
45     }
46     else {
47         int randomIndex = myRandom.nextInt(follows.size());
48         return follows.get(randomIndex);
49     }
50 }
```

Which one of the following is true regarding `getRandomNextWord` when the models have been trained on a text of N words?

A. The code has the same big-Oh complexity in both `HashMarkovModel` and `BaseMarkovModel`

B. The code in `BaseMarkovModel` is $O(N)$ and the code in `HashMarkovModel` is $O(1)$

C. The code in `HashMarkovModel` is $O(N)$ and the code in `BaseMarkovModel` is $O(1)$

D. The code in `BaseMarkovModel` is $O(N)$ and the code in `HashMarkovModel` is $O(\log N)$

PROBLEM 10:

In `HashMarkovModel`, which of the following best characterizes the Strings stored in the `List<String>` returned by the call `myMap.get(wg)` where `wg` is a `WordGram`? The only call is in the code shown below.

```
28      @Override
29      public List<String> getFollows(WordGram wgram) {
30          if (myMap.containsKey(wgram)) {
31              return myMap.get(wgram);
32          }
33          return new ArrayList<>();
34      }
```

A. The Strings are those that follow an occurrence of `wg` in the training text for the model.

B. The Strings are those that occur at indexes $k, 2k, 3k, ...$ in the array `myWords` where k is the order of the model.

C. The Strings are those that are essentially removed each time `wg.shiftAdd(s)` is called to create a new `WordGram`.

PROBLEM 11:

Both `BaseMarkovModel` and `HashMarkovModel` include the code below in the `setTraining` method.

```
myWords = text.split("\\s+");
```

This one line of code is the *only* line in `BaseMarkMode.setTraining`, but is followed by code to populate a `HashMap<WordGram,List<String>>` object in `HashMarkovModel.setTraining` as shown:

```
15      @Override
16      public void setTraining(String text) {
17          myWords = text.split("\\s+");
18          myMap.clear();
19          WordGram current = new WordGram(myWords, 0, myOrder);
20          for(int k=myOrder; k < myWords.length; k++) {
21              String next = myWords[k];
22              myMap.putIfAbsent(current,new ArrayList<>());
23              myMap.get(current).add(next);
24              current = current.shiftAdd(next);
25          }
```

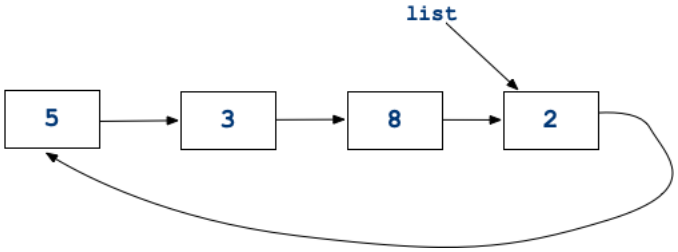
Which one statement is true?

A. Calling `setTraining(str)` has the same big-Oh complexity in both `HashMarkovModel` and `BaseMarkovModel`, essentially the number of space-separated strings in `str`.

B. Calling `setTraining(str)` has different big-Oh complexity in the different models: the call is *more efficient* in `BaseMarkovModel`.

C. Calling `setTraining(str)` has different big-Oh complexity in the different models: the call is *more efficient* in `HashMarkovModel`.

In a *circularly-linked* list, the last node of the list points to the first node rather than pointing at `null`. Such lists are typically represented by a pointer to the last node as shown, so that `list` references the last node and `list.next` references the first node. The diagram below represents the list `[5,3,8,2]` with 2 being the last node and 5 the firstnode.



The method `makeCircular` below returns a pointer to the last node of a circulary-linked list so that the call `ListNode list = makeCircular(new int[]{5,3,8,2});` returns a pointer to the last node of the list pictured above.

```
15      public ListNode makeCircular(int[] a){
16
17          ListNode first = null;
18          ListNode last = null;
19
20          for(int k=a.length-1; k >= 0; k--){
21              first = new ListNode(a[k],first);
22              if (last == null){
23                  last = first;
24              }
25          }
26          last.next = first;
27          return last;
28      }
```

PROBLEM 12:

In the code above, if parameter `a` has N values, so `a.length == N`, how many times is line 23 executed?

A. never

B. one time

C. $O(1)$ times, but greater than one

D. $O(N)$ times

PROBLEM 13:

What is returned by the call `makeCircular(new int[]{})`? Note that the parameter passed to `makeCircular` is an array with zero-length.

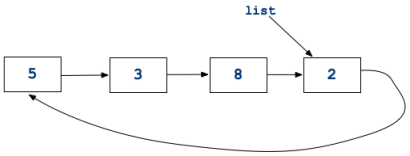
A. null

B. nothing, a null pointer exception is thrown

C. a list with one node storing a zero

PROBLEM 14:

The method `circular2String` below correctly returns a String representing its circularly-linked list parameter. The call `circular2String(list)` for the list at the beginning of this problem (and reproduced below) returns `"5 3 8 2"`.




```
30    public String circular2String(ListNode list){
31
32        if (list == null) return "";
33
34        ListNode anchor = list;
35        list = list.next;        // now first node
36
37        StringBuilder sb = new StringBuilder();
38        sb.append(""+list.info);
39
40        while (list != anchor){
41            list = list.next;
42            sb.append(" "+list.info);
43        }
44        return sb.toString().trim();
45    }
```

If the two lines 41 and 42 are interchanged, so that line 42 is `list = list.next`, what will be returned for the same call `circular2String(list)`?

A. `"5 3 8 2"`

B. `"5 5 3 8 2"`

C. `"5 5 3 8"`

PROBLEM 15:

Which the following is **a valid reason** to use a `StringBuilder` with `append` in the code above rather than a `String` with `+` to concatenate values when parameter `list` has N nodes?

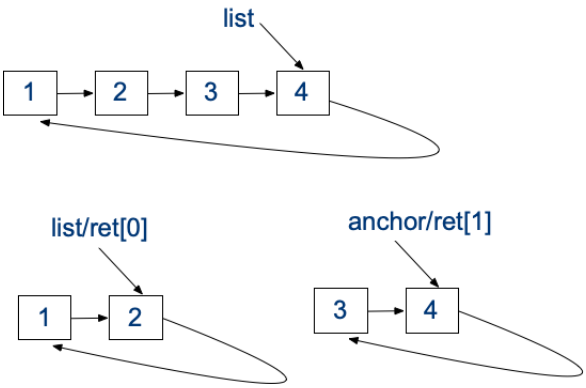
A. `StringBuilder` results in $O(N)$ time, but using a `String` would run in $O(N^2)$ time.

B. `StringBuilder` results in $O(N)$ time, and `String` also runs in $O(N)$ time, but with a larger coefficient.

The method `split` is intended to split a circularly-linked list into two circularly-linked lists, returning pointers to the last nodes of two lists by re-arranging pointers.

To return two pointers, an array with two values is returned. If parameter `list` has $2N$ nodes, the call `ListNode[] ret = split(list);` should have both `ret[0]` and `ret[1]` referencing the last nodes of two lists of N nodes; `ret[0]` is a circulary-linked list with the first half of the nodes in `list` and `ret[1]` is a circularly-linked list with the second half of the nodes.

The diagram shows what's intended for the circularly-linked list `[1,2,3,4]`



The code below works *only* for lists with an even number of nodes when two statements are added after the loop.

```
47    public ListNode[] split(ListNode list){
48
49        ListNode anchor = list;
50        list = list.next;               // first node
51        ListNode fast = list.next;      // second node
52        while (fast != anchor){
53            list = list.next;
54            fast = fast.next.next;
55        }
56        ListNode secondFirst = list.next;
57        // statement 1
58        // statement 2
59
60        return new ListNode[]{list,anchor};
61    }
```

As shown in the diagram, `list` references the node containing 2 and `anchor` references the node containing 4 when the loop exits.

PROBLEM 16:

What is the complexity of `split(list)` when `list` has N nodes if the two missing statements after the loop run in $O(1)$ time?

A. $O(\log N)$

B. $O(N)$

C. $O(N^2)$

D. $O(N \log N)$

(the code again)

```
47    public ListNode[] split(ListNode list){
48
49        ListNode anchor = list;
50        list = list.next;                // first node
51        ListNode fast = list.next;       // second node
52        while (fast != anchor){
53            list = list.next;
54            fast = fast.next.next;
55        }
56        ListNode secondFirst = list.next;
57        // statement 1
58        // statement 2
59
60        return new ListNode[]{list,anchor};
61    }
```

PROBLEM 17:

What is the first missing statement, on line 57? It should be an assignment statement and **must reference** variable `list` to create the circulary-linked list whose last node will be pointed to by `list`. **In writing code assume parameter list has an even number of nodes.**

Write the answer in the fill-in-the-blank section on the back of the answer sheet. **Bubble A for this question on the front of the answer sheet.**

PROBLEM 18:

What is the second missing statement, on line 58? It should be an assignment statement and **must reference** variables `anchor` and `secondFirst` to create the circulary-linked list whose last node will be pointed to by `anchor`. **In writing code assume parameter list has an even number of nodes.**

Write the answer in the fill-in-the-blank section on the back of the answer sheet. **Bubble A for this question on the front of the answer sheet.**

A previous APT problem called for writing the method `filter` below which removes those nodes from its linked list parameter whose values are less than their index; where the index of the first node is zero, the index of the second is 1, and so on. Lists are guaranteed to store non-negative values, so the first node is never removed.

For example, the list `2->0->3->1->4` becomes `2->3->4` since the 0 at index 1 is removed and the 1 at index 3 is removed. The node with value 3 has index 2 and the node with value 4 has index 4, so they are not removed.

The list `3->0->1->2` becomes `3`: all nodes but the first node are removed.

The code below is all green when two statements are added in the locations indicated.

```
2   public ListNode filter(ListNode list) {
3       int index = 1;
4       ListNode first = list;
5       while (list.next != null){
6           if (list.next.info < index){
7               // statement to remove
8           }
9           else {
10              // statement to advance
11          }
12          index += 1;
13      }
14      return first;
15  }
```

PROBLEM 19:

Write the code for line 7 that removes a node from the linked list in the appropriate fill-in-the blank area of the back of the bubble answer sheet. **Bubble A for this question on the front of the answer sheet.**

PROBLEM 20:

When a node is not removed, the next node in the list must be considered. Write the code for line 10, that advances the appropriate pointer so the code is all green, in the appropriate fill-in-the blank area of the back of the bubble answer sheet. **Bubble A for this question on the front of the answer sheet.**

There are three classes in P3-DNA that implement the `IDnaStrand` interface: `StringStrand`, `StringBuilderStrand`, and `LinkStrand`. The append method is shown on the right for each of these:

PROBLEM 21:

If `dna.length() == N`, which one of these methods is $O(1)$, the others have a big-Oh complexity that depends on N?

A. `StringStrand` is $O(1)$

B. `StringBuilderStrand` is $O(1)$

C. `LinkStrand` is $O(1)$

PROBLEM 22:

In the code below, `XYZ` will be replaced by one of the three stand types. Note that `dna.length() == 1000` because `dna` is aString of 1,000 `'a'` characters.

```
IDnaStrand strand = new XYZ();
String dna = "a".repeat(1000);
for(int k=0; k < 10000; k++) {
    strand.append(dna);
}
```

If the code `strand.append(dna)` executes 10,000 times where `dna.length == 1000` as shown, which of strand types executes most slowly: both with big-Oh and empirically.

A. `StringStrand`

B. `StringBuilderStrand`

C. `LinkStrand`

`StringStrand`

```
61      @Override
62      public IDnaStrand append(String dna) {
63          myInfo = myInfo + dna;
64          myAppends++;
65          return this;
66      }
```

`StringBuilderStrand`

```
59      public IDnaStrand append(String dna) {
60          myInfo.append(dna);
61          myAppends++;
62          return this;
63      }
```

`LinkStrand`

```
43      @Override
44      public IDnaStrand append(String dna) {
45          myLast.next = new Node(dna,null);
46          myLast = myLast.next;
47          myAppends += 1;
48          mySize += dna.length();
49          return this;
50      }
```

The next two problems are based on the call to `cutAndSplice` below which has *the same runtime* if the 100 on line 35 is replaced by 1000, 2000, 3000.

```
33    public void dummy(){
34        int n = 1000;
35        String splicee = "a".repeat(100);
36        IDnaStrand strand = new LinkStrand("cgat".repeat(n));
37        IDnaStrand cut = strand.cutAndSplice("cgat",splicee);
38    }
```

PROBLEM 23:

Which one of the following is true about a call `cut.toString()` if that comes after line 37?

A. It will take the same time regardless of whether line 35 uses 100, 1000,2000,3000

B. It will take more time as the value used on line 35 increases from 100, to 1000,2000,3000

PROBLEM 24:

In the code above there are 1,000 breaks/replacements of the enzyme by the splicee because of the value assigned to `n` on line 34 and used on line 36. If the 1000 is replaced by 2000, 3000, and so on how will the runtime of the call of `cutAndSplice` on line 37 change?

A. The runtime will stay the same, it does not depend on the number of breaks.

B. If there are N breaks, the runtime will be $O(N)$

C. If there are N breaks, the runtime will be $O(N^2)$

PROBLEM 25:

In the code above, if `LinkStrand` is replaced by `StringStrand` on line 36 which one of the following is true (assume the value of 1000 on line 34 and 100 on line 35 do not change).

A. The runtime of `cutAndSplice` will be about the same as when `LinkStrand` is used.

B. The runtime of `cutAndSplice` will be much greater than when `LinkStrand` is used.