CSE 1020 Introduction to Computer Science I
A sample midterm

1 (5 marks)

Draw the flow of control diagram corresponding to the following fragment.

```
statement-S;
for (initial; condition; bottom)
{
  body;
}
statement-X;
```

See Figure ? on page ? of the textbook.

2 (6 marks)

Recall that the methods `pow`(a, b) and `sqrt`(c) of the class `Math` return a^b and $\sqrt{c}$, respectively. Consider the following code snippet.

```
double x = 2.0;
output.println(x == Math.pow(Math.sqrt(x), 2.0));
```

Explain why it produces the output `false`. Recall that in mathematics, $(\sqrt{x})^2 = x$ for all $x \geq 0$.
Because real numbers are not exactly represented in Java, roundoff will cause the output `false`.

3 (6 marks)

In the table below, you find some basic constructs for regular expressions.

`[a-m]`	Range. A character between `a` and `m`, inclusive.
`[abc]`	Set. The character `a`, `b` or `c`.
`[^abc]`	Negation. Any character except `a`, `b` or `c`.
`.`	Any character.
`\d`	A digit, `[0-9]`.
`\w`	A word character, `[a-zA-Z_0-9]`.
x`?`	x, once or not at all.
x`*`	x, zero or more times.
x`+`	x, one or more times.
x{ `m, n` }	x, at least m but no more than n times.

(a) Does the string `test` match the regular expression `[te]s[t]`? If your answer is yes, explain why the string matches the regular expression. If your answer is no, explain why the string does not match the regular expression.

No. According to the regular expression, any string that matches it should start with a `t` *or an* `e`*, followed by an* `s`*. Clearly, the string* `test` *does not match that pattern.*

(b) Does the string `test` match the regular expression `([st](.{3,6}))*`? If your answer is yes, explain why the string matches the regular expression. If your answer is no, explain why the string does not match the regular expression.

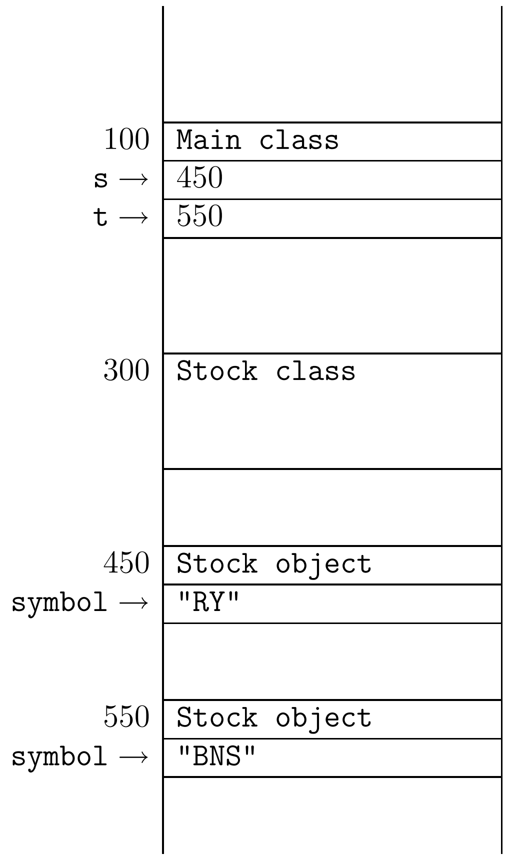
Yes. The string `test` *consists of a* `t` *followed by three characters. Hence, it consists of a* `t` *or* `s` *followed by at least three characters and at most 6 characters. Therefore, it matches* `[st](.{3,6})`*. Hence, it also matches* `([st](.{3,6}))*` *by considering the pattern* `[st](.{3,6})` *once.*

4 (18 marks)

(a) The `Stock` class has attribute `symbol`. Consider the following fragment of the `main` method.

```
Stock s = new Stock("RY");
Stock t = new Stock("BNS");
```

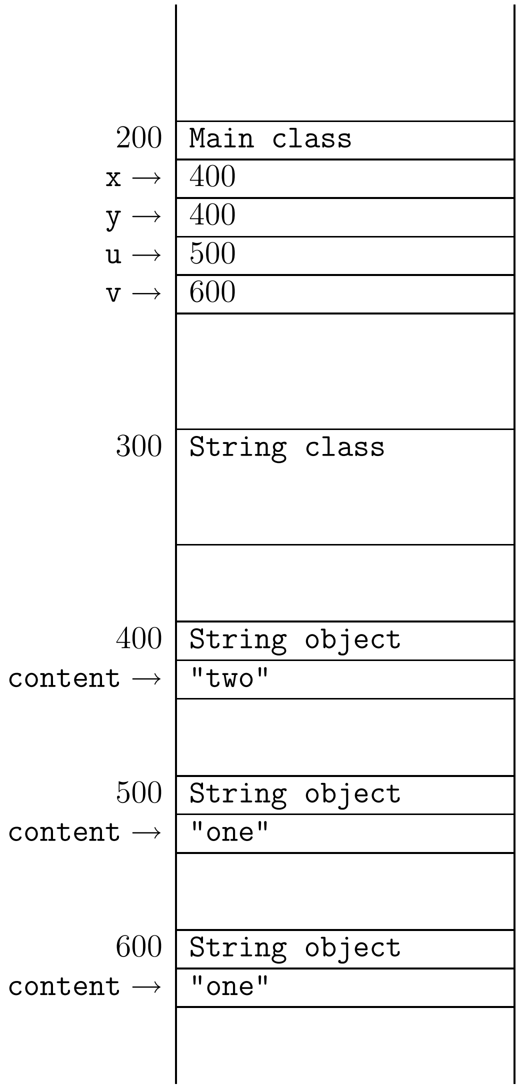
Draw the corresponding memory diagram. Make sure that the attribute `symbol` and the variables `s` and `t` are reflected in your diagram. Include both classes and objects.



(b) The `String` class has attribute `content`. Consider the following fragment of the `main` method.

```
String x = "two";
String y = "two";
String u = new String("one");
String v = new String("one");
```

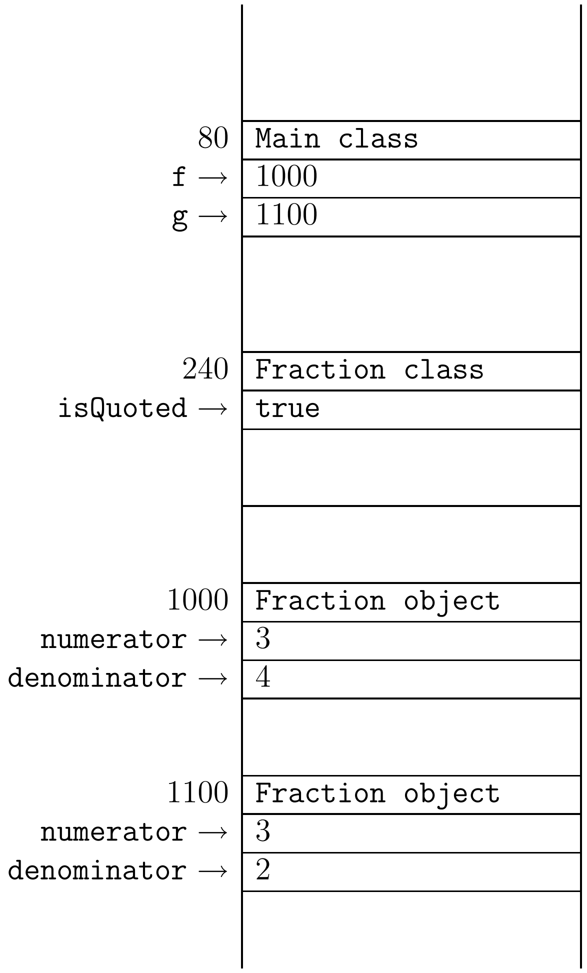
Draw the corresponding memory diagram. Make sure that the attribute `content` and the variables `x`, `y`, `u` and `v` are reflected in your diagram. Include both classes and objects.



(c) The `Fraction` class has attributes `numerator` and `denominator` and static attribute `isQuoted`. Consider the following fragment of the `main` method.

```
Fraction f = new Fraction(3, 4);
Fraction g = new Fraction(3, 2);
Fraction.isQuoted = true;
```

Draw the corresponding memory diagram. Make sure that the attributes `numerator`, `denominator` and `isQuoted` and the variables `f` and `g` are reflected in your diagram. Include both classes and objects.



5 (6 marks)

Consider the following interface.

```
char charAt(String s, int i)

Returns the i-th character of the string s.

Parameters:
 s - a string.
 i - an index.
Precondition:
 0 <= i < length of s
Returns:
 the i-th character of the string s.
Postcondition:
 the return is as stated under "Returns."
```

The `main` method of an app contains the following statement.

```
output.println(charAt("Test", 6));
```

Assume this statement causes the app to crash. Who is responsible, the client or the implementer? Explain your answer.
The client. The precondition is not satisfied. This is the client's responsibility.

6 (4 marks)

Is there any difference between having a public attribute

```
  public int age
```

and a private attribute with the following accessor and mutator.

```
  public void setAge(int age)
  public int getAge()
```

Explain your answer.
Yes. In the former case we cannot maintain the natural condition `age` $\geq$ *0, whereas we can in the latter case.*