CMPS 12A - Winter 2002
Final Exam A
March 16, 2002

Name:_____ID:_____

This is a closed note, closed book exam. Any place where you are asked to write code, you must declare all variables that you use. However, I just want code fragments, you must not write extra code such as a class specification or extraneous print statements.

Section I: 20 multiple choice questions

[5 points each] For each of the following questions, write the letter of the best answer to the right of the question including, where appropriate, “All” for “All of the above” or “None” for “None of the above”.

1. The purpose of testing is to make sure your program: All
   a. Does what it is supposed to do
   b. Doesn't do what it is not supposed to do
   c. Does what it used to do

2. Which are all Java keywords: C - array and main are not keywords
   a. int, double, for, array, and break
   b. continue, break, main, switch, and float
   c. char, boolean, final, static, and this

3. Which are all valid identifiers: C - 7eleven and data? are not valid identifiers
   a. Change, 7eleven, m77, foo_bar
   b. float, a, sIlLy, data?
   c. main, seven11, a$, test_case, java

4. Which are valid literals A - 'abc' and 4-5 are not valid literals, but 5L is
   a. 5L, 3.7, '\n', "(&(#@"
   b. 23, 4.5, -0.01, 'abc', "12897a*&)U"
   c. 5, 4-5, 0.0000004, ';', "Hello"

5. The type of a variable specifies: C
   a. its value
   b. its name
   c. the operations that are permitted on it

6. char a = 'a'; a++; a is equal to: A
   a. 'b'
   b. b
   c. error

7. Which of the following results in a = = 0
   a. int a = 3/4;
   b. int a = 3.0/4.0;
   c. double a = 3/4;

All

8. If a = 1 and b = 23, what does this expression print out

```
if(!( (1+2*a == 3) && (b-23+a != 1) ))
   System.out.println("false");
else
   System.out.println("true");
```

   a. true
   b. false
   c. nothing

B - the expression is true, so it prints out false

9. Which list of operators is listed in order of precedence
   a. +, *, /, =
   b. ++, *, +, =
   c. &&, *, +, %

B

10. What is the value of the expression

```
(3+4*5 - 4*5+3)
```

   a. 0
   b. 12
   c. 3

None - the value is 6=(3 + 20 - 20 + 3)

11. Given a = 5 and b = 7, what does this code print out

```
if (b < a)
  System.out.println("1");
else if (a + 2 == b)
  System.out.println("2");
else if (a < b)
  System.out.println("3");
```

   a. 1
   b. 2
   c. 3
   d. both 2 and 3
   e. nothing

B - the second if statement is true, and it never gets to the third

12. What does this code print out:

```
int a = 'b';
switch(a) {
  case 'a': System.out.println('1');
  case 'b': System.out.println('2');
  case 'c': System.out.println('3');
}
```

   a. 1
   b. 2
   c. 3
   d. both 2 and 3
   e. nothing

D - case 'b' is true, but there is no break, so it prints both '2' and '3'

13. What does this (ugly) code print out:

```
class Foo {
   Public static void main(String[] args) {
      int[] foo = {3, 5, 7};
      if (foo[0] > foo[1]) swap(foo, 0, 1);
      if (foo[2] < foo[1]) swap(foo, 1, 2);
      if (foo[0] > foo[1]) swap(foo, 0, 1);
      System.out.println(foo[0] + " " + foo[1] + " " + foo[2]);
   }
   void swap(int[] a, int b, int c) {
      int temp = a[b]; a[b] = a[c]; a[c] = temp;
}}
```

a. 3 5 7
b. 7 5 3
c. 15

A - the tests are false, so nothing happens to the array

14. In **for** loops:
    a. The initialization expression is executed zero or more times
    b. The boolean expression is executed zero or more times
    c. The update expression is executed zero or more times

C

15. What does this print out:

```
for(int i = 2; i < 100-i; i = i*i - i/2)
    System.out.print(i + " ");
```

a. 2 4 8 16 32
b. 2 3 7.5 56.25
c. 2 3 7 42

None

16. Which is true?
    a. return(a,b); can be used to return two values
    b. primitive types are call-by-reference
    c. the scope of an object is the class that defines its type

None

17. What does this method compute?

```
public int foo(int n) {
   if(n <= 1) return 1;
   else return 5 + foo(n-1);
}
```

a. foo(n) = the *n*th fibonacci number
b. foo(n) = 5*n
c. foo(n) = n^5
d. foo(n) = 5^n

None - it computes 5n-4

18. Method overloading is when you write a method
    a. That has too much code in it
    b. That has the same name and parameters as another method in the class
    c. That has the same name but different parameters than another method in the classl

C

19. Arrays:
    a. Once you have created an array, you can change its size
    b. The last element of an array of size *n* is at index *n* None
    c. 2D arrays are twice as big as 1D arrays

20. Public vs. Private:
    a. Public methods can only access public data elements
    b. Private data elements are accessible by methods in the same package None
    c. By default, all data elements are private

Section II: 9 questions

1. [10 points] Write code fragments to do the following:

a) Given **ints** *x*, *y*, and *z*, add *x* and *y* and store the result in *z*.

```
z = x + y;
```

b) Given a **float** called *celsius_temp*, print out "Within range" or "Out of range" depending on whether or not *celsius_temp* is greater than or equal to 0 and less than 100.

```
if(celsius_temp >= 0 && celsius_temp < 100)
    System.out.print("Within range");
else
    System.out.print("Out of range");
```

c) Given an **char** *c*, print out "A", "B.", or "C." depending on the value of *c* ('a', 'b', or 'c'). Use a **switch** statement for your solution.

```
switch(c) {
    case 'a': System.out.print("A"); break;
    case 'b': System.out.print("B"); break;
    case 'c': System.out.print("C"); break;
}
```

2. [10 points] Write a main() method that reads in four integers and prints out “yes” if the numbers were entered in increasing order and prints out “no” otherwise.

```
public static void main(String[ ] args) {
    int[ ] foo = new int[4];

    for(int i = 0; i < foo.length; i++) {
        foo[i] = Console.in.readInt();
    }

    if(foo[0] < foo[1] && foo[1] < foo[2] && foo[2] < foo[3])
        System.out.print("yes");
    else
        System.out.print("no");
}
```

3. [10 points] Write a method that tests whether the formula $a^2 + b^2 = c^2$ is true for three integers passed as parameters and returns the result of this test. Such a triple is a *Pythagorean triple* and forms a right-angle triangle with these numbers as the lengths of its sides.

```
public static boolean testPythagorean(int a, int b, int c) {
    return c*c == a*a + b*b;
}
```

4. [10 points] Write a method called *divisors*() that takes as a parameter an **int** *n*, and uses a **for** loop to find and print out the integer divisors of *n* (i.e. all integers that evenly divide into *n*). Do not use any other methods - compute the divisors yourself. *divisors*() should return the number of integer divisors of *n*.

```
public int divisors(int n) {
    int count = 0;

    for(int i = 1; i <= n; i++) {
        if(n % i == 0) {
            System.out.println(i);
            count++;
        }
    }
    return count;
}
```

5. [10 points] One way to calculate square root of a number *n* is to guess that the root is some number *x* between 0 and *n*, and then repeatedly refine your guess *x* by setting x equal to the average of *x* and *n* / *x*, until the difference between your x^2 and *n* is very small. Show how you could implement square_root() using this method to return the square root to within .0001 of a number passed as a parameter.

```
public double square_root(double n) {
    double x = n/2;

    while(Math.abs(n - x*x )> 0.0001) {
        x = (n/x + x) / 2;
    }

    return x;
}
```

6. [10 points] Write a method called pairCount() that takes as a parameter an array of characters and returns a count of the number of pairs of characters (consecutive elements of the array with the same value).

```
public int pairCount(char[ ] a) {
    int count = 0;

    for(int i = 0; i < (a.length-1); i++) {
        if(a[i] == a[i+1])
            count++;
    }

    return count;
}
```

7. [10 points] Show how you would implement the recursive method *rec*(), defined as follows:

    • *rec*(0) = 37
    • rec(1) = 73
    • *rec*(*n*) = *n* /(*rec*(*n*-1) + *rec*(*n*-2))

```
public static int rec(int n) {
    if(n == 0)
        return 37;

    if(n == 1)
        return 73;

    return n / (rec(n-1) + rec(n-2));
}
```

8. [20 points] Implement a class called Coordinate that allows the user to store an (x, y) coordinate. Provide:

a) A constructor that sets both variables to values passed as parameters.
b) An instance method called add() that adds two Coordinates and returns a new coordinate containing the sum of the other two.
c) A static method called distance() that returns the distance between two Coordinates. Recall that the distance between two coordinates (x_1, y_1) and (x_2, y_2) is $\sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$. You may use the method Math.sqrt() to calculate the square root.
d) Some test code that declares two Coordinates *A* and *B* and adds them and stores the result in *C*.

```
class Coordinate {
    private double x;
    private double y;

    Coordinate(double xval, double yval) {
        x = xval;
        y = yval;
    }

    public Coordinate add(Coordinate c) {
        return new Coordinate(x+c.x, y+c.y);
    }

    public static double distance(Coordinate c1, Coordinate c2) {
        return Math.sqrt((c2.x-c1.x)*(c2.x-c1.x) + (c2.y-c1.y)*(c2.y-c1.y));
    }
}
```

Test code:

```
Coordinate A = (1,1);
Coordinate B = (2,2);
Coordinate C = A.add(B);
```

9. [10 points] A *stack* is an abstract data structure that holds data objects of a given type.

   Stacks support two operations:
   • push(), which takes a data element and puts it on the stack, and
   • pop(), which takes a data element off of the stack and returns it.

   Stacks function as though the data elements are stacked in a pile. When a new element is pushed onto the stack, it goes on top of the pile. When an elements is popped from the stack, it is taken from the top of the pile.

   One simple way to implement a stack is to use an array to hold the data elements. A counter can keep track of how many elements are in the stack, and to find the top of the stack to add or remove data elements. When a data element is pushed onto the stack, it is put in the right spot in the array and the counter is incrememented. When a data element is popped from the stack, the counter is decremented and the appropriate item in the array is returned.

a) Write the data definitions for the Stack class, assuming that the data elements are characters.
b) Write a constructor for the Stack class
c) Write the push() method
d) Write the pop() method

```
class Stack {
    private char[ ] pile;
    private int count;

    Stack() {
        pile = new char[100];
        count = 0;
    }

    public boolean push(char c) {
        if(count < pile.length) {
            pile[count++] = c;
            return true;
        }
        return false;
    }

    public char pop() {
        if(c > 0)
            return pile[--count];
        return -1;
    }
}
```

CMPS 12A - Winter 2002
Final Exam A
March 16, 2002

Name:_____ID:_____

This is a closed note, closed book exam. Any place where you are asked to write code, you must declare all variables that you use. However, I just want code fragments, you must not write extra code such as a class specification or extraneous print statements.

Section I: 20 multiple choice questions

[5 points each] For each of the following questions, write the letter of the best answer to the right of the question including, where appropriate, “All” for “All of the above” or “None” for “None of the above”.

1. The purpose of testing is to make sure your program: All
   a. Does what it is supposed to do
   b. Doesn't do what it is not supposed to do
   c. Does what it used to do

2. Which are all Java keywords: C - array and main are not keywords
   a. int, double, for, array, and break
   b. continue, break, main, switch, and float
   c. char, boolean, final, static, and this

3. Which are all valid identifiers: C - 7eleven and data? are not valid identifiers
   a. Change, 7eleven, m77, foo_bar
   b. float, a, sIlLy, data?
   c. main, seven11, a$, test_case, java

4. Which are valid literals A - 'abc' and 4-5 are not valid literals, but 5L is
   a. 5L, 3.7, '\n', "(&(#@"
   b. 23, 4.5, -0.01, 'abc', "12897a*&)U"
   c. 5, 4-5, 0.0000004, ';', "Hello"

5. The type of a variable specifies: C
   a. its value
   b. its name
   c. the operations that are permitted on it

6. char a = 'a'; a++; a is equal to: A
   a. 'b'
   b. b
   c. error

7. Which of the following results in a = = 0
   a. int a = 3/4;
   b. int a = 3.0/4.0;
   c. double a = 3/4;

All

8. If a = 1 and b = 23, what does this expression print out

```
if(!( (1+2*a == 3) && (b-23+a != 1) ))
   System.out.println("false");
else
   System.out.println("true");
```

   a. true
   b. false
   c. nothing

B - the expression is true, so it prints out false

9. Which list of operators is listed in order of precedence
   a. +, *, /, =
   b. ++, *, +, =
   c. &&, *, +, %

B

10. What is the value of the expression

```
(3+4*5 - 4*5+3)
```

   a. 0
   b. 12
   c. 3

None - the value is 6=(3 + 20 - 20 + 3)

11. Given a = 5 and b = 7, what does this code print out

```
if (b < a)
  System.out.println("1");
else if (a + 2 == b)
  System.out.println("2");
else if (a < b)
  System.out.println("3");
```

   a. 1
   b. 2
   c. 3
   d. both 2 and 3
   e. nothing

B - the second if statement is true, and it never gets to the third

12. What does this code print out:

```
int a = 'b';
switch(a) {
  case 'a': System.out.println('1');
  case 'b': System.out.println('2');
  case 'c': System.out.println('3');
}
```

   a. 1
   b. 2
   c. 3
   d. both 2 and 3
   e. nothing

D - case 'b' is true, but there is no break, so it prints both '2' and '3'

13. What does this (ugly) code print out:

```
class Foo {
   Public static void main(String[] args) {
      int[] foo = {3, 5, 7};
      if (foo[0] > foo[1]) swap(foo, 0, 1);
      if (foo[2] < foo[1]) swap(foo, 1, 2);
      if (foo[0] > foo[1]) swap(foo, 0, 1);
      System.out.println(foo[0] + " " + foo[1] + " " + foo[2]);
   }
   void swap(int[] a, int b, int c) {
      int temp = a[b]; a[b] = a[c]; a[c] = temp;
}}
```

a. 3 5 7
b. 7 5 3
c. 15

A - the tests are false, so nothing happens to the array

14. In **for** loops:
    a. The initialization expression is executed zero or more times
    b. The boolean expression is executed zero or more times
    c. The update expression is executed zero or more times

C

15. What does this print out:

```
for(int i = 2; i < 100-i; i = i*i - i/2)
    System.out.print(i + " ");
```

a. 2 4 8 16 32
b. 2 3 7.5 56.25
c. 2 3 7 42

None

16. Which is true?
    a. return(a,b); can be used to return two values
    b. primitive types are call-by-reference
    c. the scope of an object is the class that defines its type

None

17. What does this method compute?

```
public int foo(int n) {
   if(n <= 1) return 1;
   else return 5 + foo(n-1);
}
```

a. foo(n) = the *n*th fibonacci number
b. foo(n) = 5*n
c. foo(n) = n^5
d. foo(n) = 5^n

None - it computes 5n-4

18. Method overloading is when you write a method
    a. That has too much code in it
    b. That has the same name and parameters as another method in the class
    c. That has the same name but different parameters than another method in the classl

C

19. Arrays:
    a. Once you have created an array, you can change its size
    b. The last element of an array of size *n* is at index *n* None
    c. 2D arrays are twice as big as 1D arrays

20. Public vs. Private:
    a. Public methods can only access public data elements
    b. Private data elements are accessible by methods in the same package None
    c. By default, all data elements are private

Section II: 9 questions

1. [10 points] Write code fragments to do the following:

a) Given **ints** *x*, *y*, and *z*, add *x* and *y* and store the result in *z*.

```
z = x + y;
```

b) Given a **float** called *celsius_temp*, print out "Within range" or "Out of range" depending on whether or not *celsius_temp* is greater than or equal to 0 and less than 100.

```
if(celsius_temp >= 0 && celsius_temp < 100)
    System.out.print("Within range");
else
    System.out.print("Out of range");
```

c) Given an **char** *c*, print out "A", "B.", or "C." depending on the value of *c* ('a', 'b', or 'c'). Use a **switch** statement for your solution.

```
switch(c) {
    case 'a': System.out.print("A"); break;
    case 'b': System.out.print("B"); break;
    case 'c': System.out.print("C"); break;
}
```

2. [10 points] Write a main() method that reads in four integers and prints out “yes” if the numbers were entered in increasing order and prints out “no” otherwise.

```
public static void main(String[ ] args) {
    int[ ] foo = new int[4];

    for(int i = 0; i < foo.length; i++) {
        foo[i] = Console.in.readInt();
    }

    if(foo[0] < foo[1] && foo[1] < foo[2] && foo[2] < foo[3])
        System.out.print("yes");
    else
        System.out.print("no");
}
```

3. [10 points] Write a method that tests whether the formula $a^2 + b^2 = c^2$ is true for three integers passed as parameters and returns the result of this test. Such a triple is a *Pythagorean triple* and forms a right-angle triangle with these numbers as the lengths of its sides.

```
public static boolean testPythagorean(int a, int b, int c) {
    return c*c == a*a + b*b;
}
```

4. [10 points] Write a method called *divisors*() that takes as a parameter an **int** *n*, and uses a **for** loop to find and print out the integer divisors of *n* (i.e. all integers that evenly divide into *n*). Do not use any other methods - compute the divisors yourself. *divisors*() should return the number of integer divisors of *n*.

```
public int divisors(int n) {
    int count = 0;

    for(int i = 1; i <= n; i++) {
        if(n % i == 0) {
            System.out.println(i);
            count++;
        }
    }
    return count;
}
```

5. [10 points] One way to calculate square root of a number *n* is to guess that the root is some number *x* between 0 and *n*, and then repeatedly refine your guess *x* by setting x equal to the average of *x* and *n* / *x*, until the difference between your x^2 and *n* is very small. Show how you could implement square_root() using this method to return the square root to within .0001 of a number passed as a parameter.

```
public double square_root(double n) {
    double x = n/2;

    while(Math.abs(n - x*x )> 0.0001) {
        x = (n/x + x) / 2;
    }

    return x;
}
```

6. [10 points] Write a method called pairCount() that takes as a parameter an array of characters and returns a count of the number of pairs of characters (consecutive elements of the array with the same value).

```
public int pairCount(char[ ] a) {
    int count = 0;

    for(int i = 0; i < (a.length-1); i++) {
        if(a[i] == a[i+1])
            count++;
    }

    return count;
}
```

7. [10 points] Show how you would implement the recursive method *rec*(), defined as follows:

    • *rec*(0) = 37
    • rec(1) = 73
    • *rec*(*n*) = *n* /(*rec*(*n*-1) + *rec*(*n*-2))

```
public static int rec(int n) {
    if(n == 0)
        return 37;

    if(n == 1)
        return 73;

    return n / (rec(n-1) + rec(n-2));
}
```

8. [20 points] Implement a class called Coordinate that allows the user to store an (x, y) coordinate. Provide:

a) A constructor that sets both variables to values passed as parameters.
b) An instance method called add() that adds two Coordinates and returns a new coordinate containing the sum of the other two.
c) A static method called distance() that returns the distance between two Coordinates. Recall that the distance between two coordinates (x_1, y_1) and (x_2, y_2) is $\sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$. You may use the method Math.sqrt() to calculate the square root.
d) Some test code that declares two Coordinates *A* and *B* and adds them and stores the result in *C*.

```
class Coordinate {
    private double x;
    private double y;

    Coordinate(double xval, double yval) {
        x = xval;
        y = yval;
    }

    public Coordinate add(Coordinate c) {
        return new Coordinate(x+c.x, y+c.y);
    }

    public static double distance(Coordinate c1, Coordinate c2) {
        return Math.sqrt((c2.x-c1.x)*(c2.x-c1.x) + (c2.y-c1.y)*(c2.y-c1.y));
    }
}
```

Test code:

```
Coordinate A = (1,1);
Coordinate B = (2,2);
Coordinate C = A.add(B);
```

9. [10 points] A *stack* is an abstract data structure that holds data objects of a given type.

   Stacks support two operations:
   • push(), which takes a data element and puts it on the stack, and
   • pop(), which takes a data element off of the stack and returns it.

   Stacks function as though the data elements are stacked in a pile. When a new element is pushed onto the stack, it goes on top of the pile. When an elements is popped from the stack, it is taken from the top of the pile.

   One simple way to implement a stack is to use an array to hold the data elements. A counter can keep track of how many elements are in the stack, and to find the top of the stack to add or remove data elements. When a data element is pushed onto the stack, it is put in the right spot in the array and the counter is incrememented. When a data element is popped from the stack, the counter is decremented and the appropriate item in the array is returned.

a) Write the data definitions for the Stack class, assuming that the data elements are characters.
b) Write a constructor for the Stack class
c) Write the push() method
d) Write the pop() method

```
class Stack {
    private char[ ] pile;
    private int count;

    Stack() {
        pile = new char[100];
        count = 0;
    }

    public boolean push(char c) {
        if(count < pile.length) {
            pile[count++] = c;
            return true;
        }
        return false;
    }

    public char pop() {
        if(c > 0)
            return pile[--count];
        return -1;
    }
}
```

CMPS 12A - Winter 2002
Final Exam A
March 16, 2002

Name:_____ID:_____

This is a closed note, closed book exam. Any place where you are asked to write code, you must declare all variables that you use. However, I just want code fragments, you must not write extra code such as a class specification or extraneous print statements.

Section I: 20 multiple choice questions

[5 points each] For each of the following questions, write the letter of the best answer to the right of the question including, where appropriate, “All” for “All of the above” or “None” for “None of the above”.

1. The purpose of testing is to make sure your program: All
   a. Does what it is supposed to do
   b. Doesn't do what it is not supposed to do
   c. Does what it used to do

2. Which are all Java keywords: C - array and main are not keywords
   a. int, double, for, array, and break
   b. continue, break, main, switch, and float
   c. char, boolean, final, static, and this

3. Which are all valid identifiers: C - 7eleven and data? are not valid identifiers
   a. Change, 7eleven, m77, foo_bar
   b. float, a, sIlLy, data?
   c. main, seven11, a$, test_case, java

4. Which are valid literals A - 'abc' and 4-5 are not valid literals, but 5L is
   a. 5L, 3.7, '\n', "(&(#@"
   b. 23, 4.5, -0.01, 'abc', "12897a*&)U"
   c. 5, 4-5, 0.0000004, ';', "Hello"

5. The type of a variable specifies: C
   a. its value
   b. its name
   c. the operations that are permitted on it

6. char a = 'a'; a++; a is equal to: A
   a. 'b'
   b. b
   c. error

7. Which of the following results in a = = 0
   a. int a = 3/4;
   b. int a = 3.0/4.0;
   c. double a = 3/4;

All

8. If a = 1 and b = 23, what does this expression print out

```
if(!( (1+2*a == 3) && (b-23+a != 1) ))
   System.out.println("false");
else
   System.out.println("true");
```

   a. true
   b. false
   c. nothing

B - the expression is true, so it prints out false

9. Which list of operators is listed in order of precedence
   a. +, *, /, =
   b. ++, *, +, =
   c. &&, *, +, %

B

10. What is the value of the expression

```
(3+4*5 - 4*5+3)
```

   a. 0
   b. 12
   c. 3

None - the value is 6=(3 + 20 - 20 + 3)

11. Given a = 5 and b = 7, what does this code print out

```
if (b < a)
  System.out.println("1");
else if (a + 2 == b)
  System.out.println("2");
else if (a < b)
  System.out.println("3");
```

   a. 1
   b. 2
   c. 3
   d. both 2 and 3
   e. nothing

B - the second if statement is true, and it never gets to the third

12. What does this code print out:

```
int a = 'b';
switch(a) {
  case 'a': System.out.println('1');
  case 'b': System.out.println('2');
  case 'c': System.out.println('3');
}
```

   a. 1
   b. 2
   c. 3
   d. both 2 and 3
   e. nothing

D - case 'b' is true, but there is no break, so it prints both '2' and '3'

13. What does this (ugly) code print out:

```
class Foo {
   Public static void main(String[] args) {
      int[] foo = {3, 5, 7};
      if (foo[0] > foo[1]) swap(foo, 0, 1);
      if (foo[2] < foo[1]) swap(foo, 1, 2);
      if (foo[0] > foo[1]) swap(foo, 0, 1);
      System.out.println(foo[0] + " " + foo[1] + " " + foo[2]);
   }
   void swap(int[] a, int b, int c) {
      int temp = a[b]; a[b] = a[c]; a[c] = temp;
}}
```

a. 3 5 7
b. 7 5 3
c. 15

A - the tests are false, so nothing happens to the array

14. In **for** loops:
    a. The initialization expression is executed zero or more times
    b. The boolean expression is executed zero or more times
    c. The update expression is executed zero or more times

C

15. What does this print out:

```
for(int i = 2; i < 100-i; i = i*i - i/2)
    System.out.print(i + " ");
```

a. 2 4 8 16 32
b. 2 3 7.5 56.25
c. 2 3 7 42

None

16. Which is true?
    a. return(a,b); can be used to return two values
    b. primitive types are call-by-reference
    c. the scope of an object is the class that defines its type

None

17. What does this method compute?

```
public int foo(int n) {
   if(n <= 1) return 1;
   else return 5 + foo(n-1);
}
```

a. foo(n) = the *n*th fibonacci number
b. foo(n) = 5*n
c. foo(n) = n^5
d. foo(n) = 5^n

None - it computes 5n-4

18. Method overloading is when you write a method
    a. That has too much code in it
    b. That has the same name and parameters as another method in the class
    c. That has the same name but different parameters than another method in the classl

C

19. Arrays:
    a. Once you have created an array, you can change its size
    b. The last element of an array of size *n* is at index *n* None
    c. 2D arrays are twice as big as 1D arrays

20. Public vs. Private:
    a. Public methods can only access public data elements
    b. Private data elements are accessible by methods in the same package None
    c. By default, all data elements are private

Section II: 9 questions

1. [10 points] Write code fragments to do the following:

a) Given **ints** *x*, *y*, and *z*, add *x* and *y* and store the result in *z*.

```
z = x + y;
```

b) Given a **float** called *celsius_temp*, print out "Within range" or "Out of range" depending on whether or not *celsius_temp* is greater than or equal to 0 and less than 100.

```
if(celsius_temp >= 0 && celsius_temp < 100)
    System.out.print("Within range");
else
    System.out.print("Out of range");
```

c) Given an **char** *c*, print out "A", "B.", or "C." depending on the value of *c* ('a', 'b', or 'c'). Use a **switch** statement for your solution.

```
switch(c) {
    case 'a': System.out.print("A"); break;
    case 'b': System.out.print("B"); break;
    case 'c': System.out.print("C"); break;
}
```

2. [10 points] Write a main() method that reads in four integers and prints out “yes” if the numbers were entered in increasing order and prints out “no” otherwise.

```
public static void main(String[ ] args) {
    int[ ] foo = new int[4];

    for(int i = 0; i < foo.length; i++) {
        foo[i] = Console.in.readInt();
    }

    if(foo[0] < foo[1] && foo[1] < foo[2] && foo[2] < foo[3])
        System.out.print("yes");
    else
        System.out.print("no");
}
```

3. [10 points] Write a method that tests whether the formula $a^2 + b^2 = c^2$ is true for three integers passed as parameters and returns the result of this test. Such a triple is a *Pythagorean triple* and forms a right-angle triangle with these numbers as the lengths of its sides.

```
public static boolean testPythagorean(int a, int b, int c) {
    return c*c == a*a + b*b;
}
```

4. [10 points] Write a method called *divisors*() that takes as a parameter an **int** *n*, and uses a **for** loop to find and print out the integer divisors of *n* (i.e. all integers that evenly divide into *n*). Do not use any other methods - compute the divisors yourself. *divisors*() should return the number of integer divisors of *n*.

```
public int divisors(int n) {
    int count = 0;

    for(int i = 1; i <= n; i++) {
        if(n % i == 0) {
            System.out.println(i);
            count++;
        }
    }
    return count;
}
```

5. [10 points] One way to calculate square root of a number *n* is to guess that the root is some number *x* between 0 and *n*, and then repeatedly refine your guess *x* by setting x equal to the average of *x* and *n* / *x*, until the difference between your x^2 and *n* is very small. Show how you could implement square_root() using this method to return the square root to within .0001 of a number passed as a parameter.

```
public double square_root(double n) {
    double x = n/2;

    while(Math.abs(n - x*x )> 0.0001) {
        x = (n/x + x) / 2;
    }

    return x;
}
```

6. [10 points] Write a method called pairCount() that takes as a parameter an array of characters and returns a count of the number of pairs of characters (consecutive elements of the array with the same value).

```
public int pairCount(char[ ] a) {
    int count = 0;

    for(int i = 0; i < (a.length-1); i++) {
        if(a[i] == a[i+1])
            count++;
    }

    return count;
}
```

7. [10 points] Show how you would implement the recursive method *rec*(), defined as follows:

    • *rec*(0) = 37
    • rec(1) = 73
    • *rec*(*n*) = *n* /(*rec*(*n*-1) + *rec*(*n*-2))

```
public static int rec(int n) {
    if(n == 0)
        return 37;

    if(n == 1)
        return 73;

    return n / (rec(n-1) + rec(n-2));
}
```

8. [20 points] Implement a class called Coordinate that allows the user to store an (x, y) coordinate. Provide:

a) A constructor that sets both variables to values passed as parameters.
b) An instance method called add() that adds two Coordinates and returns a new coordinate containing the sum of the other two.
c) A static method called distance() that returns the distance between two Coordinates. Recall that the distance between two coordinates (x_1, y_1) and (x_2, y_2) is $\sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$. You may use the method Math.sqrt() to calculate the square root.
d) Some test code that declares two Coordinates *A* and *B* and adds them and stores the result in *C*.

```
class Coordinate {
    private double x;
    private double y;

    Coordinate(double xval, double yval) {
        x = xval;
        y = yval;
    }

    public Coordinate add(Coordinate c) {
        return new Coordinate(x+c.x, y+c.y);
    }

    public static double distance(Coordinate c1, Coordinate c2) {
        return Math.sqrt((c2.x-c1.x)*(c2.x-c1.x) + (c2.y-c1.y)*(c2.y-c1.y));
    }
}
```

Test code:

```
Coordinate A = (1,1);
Coordinate B = (2,2);
Coordinate C = A.add(B);
```

9. [10 points] A *stack* is an abstract data structure that holds data objects of a given type.

   Stacks support two operations:
   • push(), which takes a data element and puts it on the stack, and
   • pop(), which takes a data element off of the stack and returns it.

   Stacks function as though the data elements are stacked in a pile. When a new element is pushed onto the stack, it goes on top of the pile. When an elements is popped from the stack, it is taken from the top of the pile.

   One simple way to implement a stack is to use an array to hold the data elements. A counter can keep track of how many elements are in the stack, and to find the top of the stack to add or remove data elements. When a data element is pushed onto the stack, it is put in the right spot in the array and the counter is incrememented. When a data element is popped from the stack, the counter is decremented and the appropriate item in the array is returned.

a) Write the data definitions for the Stack class, assuming that the data elements are characters.
b) Write a constructor for the Stack class
c) Write the push() method
d) Write the pop() method

```
class Stack {
    private char[ ] pile;
    private int count;

    Stack() {
        pile = new char[100];
        count = 0;
    }

    public boolean push(char c) {
        if(count < pile.length) {
            pile[count++] = c;
            return true;
        }
        return false;
    }

    public char pop() {
        if(c > 0)
            return pile[--count];
        return -1;
    }
}
```