CMPS 12A - Winter 2002
Final Exam B
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. 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   **B**
   c. 3
   d. both 2 and 3
   e. nothing

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

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

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

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

1

6. 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");
```
   B - the test is true, so false is printed out
   a. true
   b. false
   c. nothing

7. 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

8. Method overloading is when you write a method   *C*
   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 class

9. What is the value of the expression
```
(3+4*5 - 4*5+3)
```
   None - it print out 6 (= 3 + 20 - 20 + 3)
   a. 0
   b. 12
   c. 3

10. 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');
}
```
   D - no break statements
   a. 1
   b. 2
   c. 3
   d. both 2 and 3
   e. nothing

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

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

13. Public vs. Private:
    a. Public methods can only access public data elements
    b. Private data elements are accessible by methods in the same package *C*
    c. By default, all data elements are private

14. 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

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

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

16. 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   *None*
    c. 2 3 7 42

17. Which is true?
    a. return(a,b); can be used to return two values
    b. primitive types are call-by-reference   *None*
    c. the scope of an object is the class that defines its type

18. 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   *None*
    b. foo(n) = 5*n
    c. foo(n) = n^5
    d. foo(n) = 5^n

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*
    c. 2D arrays are twice as big as 1D arrays

None

20. Which are all valid identifiers:
    a. Change, 7eleven, m77, foo_bar
    b. float, a, sIlLy, data?
    c. main, seven11, a$, test_case, java

C - 7eleven and data? are invalid

Section II: 9 questions

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

a) Given **ints** *a*, *b*, and *c*, add *a* and *b* and store the result in *c*.

```
c = a + b;
```

b) Given a **double** called *time*, print out "Open" or "Closed" depending on whether or not *time* is greater than or equal to 800 and less than 1700.

```
if(time >= 800 && time < 1700)
    System.out.print("Open");
else
    System.out.print("Closed");
```

c) Given an **char** *c*, print out "1", "2", or "3" depending on the value of *c* ('a', 'b', or 'c'). Use a **switch** statement for your solution.

```
switch(c) {
    case 'a': System.out.print("1"); break;
    case 'b': System.out.print("2"); break;
    case 'c': System.out.print("3"); break;
}
```

2. [10 points] Write a method called *divisors*() that takes as a parameter an **int** *k*, and uses a **for** loop to find and print out the integer divisors of *k* (i.e. all integers that evenly divide into *k*). Do not use any other methods - compute the divisors yourself. *divisors*() should return the number of integer divisors of *k*.

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

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

3. [10 points] One way to calculate square root of a number *a* is to guess that the root is some number *n* between 0 and *a*, and then repeatedly refine your guess *n* by setting n equal to the average of *n* and *a* / *n*, until the difference between your n^2 and *a* is very small. Show how you could implement sqrt() using this method to return the square root to within .00001 of a number passed as a parameter.

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

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

    return n;
}
```

4. [10 points] Write a main() method that reads in four integers and prints out "increasing" if the numbers were entered in increasing order and prints out "not increasing" 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("Increasing");
    else
        System.out.print("Not Increasing");
}
```

5. [10 points] Write a method that tests whether the formula $x^2 - y^2 = z^2$ is true for three integers passed as parameters and returns the result of this test.

```
public static boolean testPythagorean(int x, int y, int z) {
    return x*x - y*y == z*z;
}
```

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:

- $rec(0) = 73$
- $rec(n) = n * rec(n-1) + rec(n-2)$

```
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 Coord 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 Coords and returns a new coordinate containing the sum of the other two.
c) A static method called distance() that returns the distance between two Coords. 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 Coords X and Y and adds them and stores the result in Z.

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

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

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

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

Test code:
Coord X = (1,1);
Coord Y = (2,2);
Coord Z = X.add(Y);
```

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 queue, 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;
    }
}
```

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. 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   B
    c. 3
    d. both 2 and 3
    e. nothing

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

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

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

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

6. 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 test is true, so false is printed out

7. Which are all Java keywords:
    a.   int, double, for, array, and break
    b.   continue, break, main, switch, and float
    c.   char, boolean, final, static, and this

C - array and main are not keywords

8. 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 class

C

9. What is the value of the expression

```
(3+4*5 - 4*5+3)
```
    a.   0
    b.   12
    c.   3

None - it print out 6 (= 3 + 20 - 20 + 3)

10. 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 - no break statements

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

A

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

C

13. Public vs. Private:
    a. Public methods can only access public data elements
    b. Private data elements are accessible by methods in the same package *C*
    c. By default, all data elements are private

14. 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

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

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

16. 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   *None*
    c. 2 3 7 42

17. Which is true?
    a. return(a,b); can be used to return two values
    b. primitive types are call-by-reference   *None*
    c. the scope of an object is the class that defines its type

18. 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 nth fibonacci number   *None*
    b. foo(n) = 5*n
    c. foo(n) = n^5
    d. foo(n) = 5^n

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*
    c. 2D arrays are twice as big as 1D arrays

    None

20. Which are all valid identifiers:
    a. Change, 7eleven, m77, foo_bar
    b. float, a, sIlLy, data?
    c. main, seven11, a$, test_case, java

    C - 7eleven and data? are invalid

Section II: 9 questions

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

a) Given **ints** *a*, *b*, and *c*, add *a* and *b* and store the result in *c*.

    c = a + b;

b) Given a **double** called *time*, print out "Open" or "Closed" depending on whether or not *time* is greater than or equal to 800 and less than 1700.

    if(time >= 800 && time < 1700)
        System.out.print("Open");
    else
        System.out.print("Closed");

c) Given an **char** *c*, print out "1", "2", or "3" depending on the value of *c* ('a', 'b', or 'c'). Use a **switch** statement for your solution.

    switch(c) {
        case 'a': System.out.print("1"); break;
        case 'b': System.out.print("2"); break;
        case 'c': System.out.print("3"); break;
    }

4

2. [10 points] Write a method called *divisors*() that takes as a parameter an **int** k, and uses a **for** loop to find and print out the integer divisors of k (i.e. all integers that evenly divide into k). Do not use any other methods - compute the divisors yourself. *divisors*() should return the number of integer divisors of k.

```java
public int divisors(int k) {
    int count = 0;

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

3. [10 points] One way to calculate square root of a number a is to guess that the root is some number n between 0 and a, and then repeatedly refine your guess n by setting n equal to the average of n and a / n, until the difference between your n^2 and a is very small. Show how you could implement sqrt() using this method to return the square root to within .00001 of a number passed as a parameter.

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

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

    return n;
}
```

4. [10 points] Write a main() method that reads in four integers and prints out "increasing" if the numbers were entered in increasing order and prints out "not increasing" otherwise.

```java
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("Increasing");
    else
        System.out.print("Not Increasing");
}
```

5. [10 points] Write a method that tests whether the formula $x^2 - y^2 = z^2$ is true for three integers passed as parameters and returns the result of this test.

```java
public static boolean testPythagorean(int x, int y, int z) {
    return x*x - y*y == z*z;
}
```

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:

- $rec(0) = 73$
- $rec(n) = n * rec(n-1) + rec(n-2)$

```
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 Coord 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 Coords and returns a new coordinate containing the sum of the other two.
c) A static method called distance() that returns the distance between two Coords. 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 Coords X and Y and adds them and stores the result in Z.

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

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

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

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

Test code:
Coord X = (1,1);
Coord Y = (2,2);
Coord Z = X.add(Y);
```

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 queue, 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 B
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. 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   B
    c. 3
    d. both 2 and 3
    e. nothing

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

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

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

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

1

6. 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");
```
   B - the test is true, so false is printed out
   a. true
   b. false
   c. nothing

7. 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

8. Method overloading is when you write a method   *C*
   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 class

9. What is the value of the expression
   `(3+4*5 - 4*5+3)`   *None - it print out 6 (= 3 + 20 - 20 + 3)*
   a. 0
   b. 12
   c. 3

10. 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');
}
```
   D - no break statements
   a. 1
   b. 2
   c. 3
   d. both 2 and 3
   e. nothing

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

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

13. Public vs. Private:
    a. Public methods can only access public data elements
    b. Private data elements are accessible by methods in the same package *C*
    c. By default, all data elements are private

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

```
class Foo {
   Public static void main(String[] args) {
      int[] foo = {3, 5, 7};                                    A
      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

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

16. 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   *None*
   c. 2 3 7 42

17. Which is true?
    a. return(a,b); can be used to return two values
    b. primitive types are call-by-reference   *None*
    c. the scope of an object is the class that defines its type

18. 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   *None*
   b. foo(n) = 5*n
   c. foo(n) = n^5
   d. foo(n) = 5^n

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*
    c. 2D arrays are twice as big as 1D arrays

None

20. Which are all valid identifiers:
    a. Change, 7eleven, m77, foo_bar
    b. float, a, sIlLy, data?
    c. main, seven11, a$, test_case, java

C - 7eleven and data? are invalid

Section II: 9 questions

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

a) Given **ints** *a*, *b*, and *c*, add *a* and *b* and store the result in *c*.

```
c = a + b;
```

b) Given a **double** called *time*, print out "Open" or "Closed" depending on whether or not *time* is greater than or equal to 800 and less than 1700.

```
if(time >= 800 && time < 1700)
    System.out.print("Open");
else
    System.out.print("Closed");
```

c) Given an **char** *c*, print out "1", "2", or "3" depending on the value of *c* ('a', 'b', or 'c'). Use a **switch** statement for your solution.

```
switch(c) {
    case 'a': System.out.print("1"); break;
    case 'b': System.out.print("2"); break;
    case 'c': System.out.print("3"); break;
}
```

2. [10 points] Write a method called *divisors()* that takes as a parameter an **int** k, and uses a **for** loop to find and print out the integer divisors of k (i.e. all integers that evenly divide into k). Do not use any other methods - compute the divisors yourself. *divisors()* should return the number of integer divisors of k.

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

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

3. [10 points] One way to calculate square root of a number *a* is to guess that the root is some number *n* between 0 and *a*, and then repeatedly refine your guess *n* by setting n equal to the average of *n* and *a* / *n*, until the difference between your n^2 and *a* is very small. Show how you could implement sqrt() using this method to return the square root to within .00001 of a number passed as a parameter.

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

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

    return n;
}
```

4. [10 points] Write a main() method that reads in four integers and prints out "increasing" if the numbers were entered in increasing order and prints out "not increasing" otherwise.

```java
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("Increasing");
    else
        System.out.print("Not Increasing");
}
```

5. [10 points] Write a method that tests whether the formula $x^2 - y^2 = z^2$ is true for three integers passed as parameters and returns the result of this test.

```java
public static boolean testPythagorean(int x, int y, int z) {
    return x*x - y*y == z*z;
}
```

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:

- $rec(0) = 73$
- $rec(n) = n * rec(n-1) + rec(n-2)$

```
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 Coord 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 Coords and returns a new coordinate containing the sum of the other two.
c) A static method called distance() that returns the distance between two Coords. 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 Coords X and Y and adds them and stores the result in Z.

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

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

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

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

Test code:
Coord X = (1,1);
Coord Y = (2,2);
Coord Z = X.add(Y);
```

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 queue, 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 incremememted. 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 B
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. 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   B
    c. 3
    d. both 2 and 3
    e. nothing

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

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

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

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

1

6. 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 test is true, so false is printed out

7. Which are all Java keywords:
   a. int, double, for, array, and break
   b. continue, break, main, switch, and float
   c. char, boolean, final, static, and this

C - array and main are not keywords

8. 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 class

C

9. What is the value of the expression
   ```
   (3+4*5 - 4*5+3)
   ```
   a. 0
   b. 12
   c. 3

None - it print out 6 (= 3 + 20 - 20 + 3)

10. 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 - no break statements

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

A

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

C

13. Public vs. Private:
    a. Public methods can only access public data elements
    b. Private data elements are accessible by methods in the same package C
    c. By default, all data elements are private

14. What does this (ugly) code print out:
```
class Foo {
    Public static void main(String[] args) {
        int[] foo = {3, 5, 7};                                    A
        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

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

16. 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   None
    c. 2 3 7 42

17. Which is true?
    a. return(a,b); can be used to return two values
    b. primitive types are call-by-reference   None
    c. the scope of an object is the class that defines its type

18. 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 nth fibonacci number   None
    b. foo(n) = 5*n
    c. foo(n) = n^5
    d. foo(n) = 5^n

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*
    c. 2D arrays are twice as big as 1D arrays

None

20. Which are all valid identifiers:
    a. Change, 7eleven, m77, foo_bar
    b. float, a, sIlLy, data?
    c. main, seven11, a$, test_case, java

C - 7eleven and data? are invalid

Section II: 9 questions

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

a) Given **ints** *a*, *b*, and *c*, add *a* and *b* and store the result in *c*.

<span style="color:red">

```
c = a + b;
```


b) Given a **double** called *time*, print out "Open" or "Closed" depending on whether or not *time* is greater than or equal to 800 and less than 1700.

<span style="color:red">

```
if(time >= 800 && time < 1700)
    System.out.print("Open");
else
    System.out.print("Closed");
```


c) Given an **char** *c*, print out "1", "2", or "3" depending on the value of *c* ('a', 'b', or 'c'). Use a **switch** statement for your solution.

<span style="color:red">

```
switch(c) {
    case 'a': System.out.print("1"); break;
    case 'b': System.out.print("2"); break;
    case 'c': System.out.print("3"); break;
}
```


4

2. [10 points] Write a method called *divisors*() that takes as a parameter an **int** k, and uses a **for** loop to find and print out the integer divisors of k (i.e. all integers that evenly divide into k). Do not use any other methods - compute the divisors yourself. *divisors*() should return the number of integer divisors of k.

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

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

3. [10 points] One way to calculate square root of a number *a* is to guess that the root is some number *n* between 0 and *a*, and then repeatedly refine your guess *n* by setting n equal to the average of *n* and *a* / *n*, until the difference between your n^2 and *a* is very small. Show how you could implement sqrt() using this method to return the square root to within .00001 of a number passed as a parameter.

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

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

    return n;
}
```

4. [10 points] Write a main() method that reads in four integers and prints out "increasing" if the numbers were entered in increasing order and prints out "not increasing" otherwise.

```java
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("Increasing");
    else
        System.out.print("Not Increasing");
}
```

5. [10 points] Write a method that tests whether the formula $x^2 - y^2 = z^2$ is true for three integers passed as parameters and returns the result of this test.

```java
public static boolean testPythagorean(int x, int y, int z) {
    return x*x - y*y == z*z;
}
```

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:

- $rec(0) = 73$
- $rec(n) = n * rec(n-1) + rec(n-2)$

```
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 Coord 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 Coords and returns a new coordinate containing the sum of the other two.
c) A static method called distance() that returns the distance between two Coords. 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 Coords X and Y and adds them and stores the result in Z.

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

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

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

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

Test code:
Coord X = (1,1);
Coord Y = (2,2);
Coord Z = X.add(Y);
```

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 queue, 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 B
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. 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   **B**
   c. 3
   d. both 2 and 3
   e. nothing

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

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

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

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

6. 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");
```

    B - the test is true, so false is printed out

   a. true
   b. false
   c. nothing

7. 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

8. Method overloading is when you write a method   *C*
   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 class

9. What is the value of the expression

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

    None - it print out 6 (= 3 + 20 - 20 + 3)

   a. 0
   b. 12
   c. 3

10. 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');
}
```

    D - no break statements

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

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

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

13. Public vs. Private:
    a. Public methods can only access public data elements
    b. Private data elements are accessible by methods in the same package *C*
    c. By default, all data elements are private

14. 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
    a. 3 5 7
    b. 7 5 3
    c. 15

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

16. 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   *None*
    c. 2 3 7 42

17. Which is true?
    a. return(a,b); can be used to return two values
    b. primitive types are call-by-reference   *None*
    c. the scope of an object is the class that defines its type

18. 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   *None*
    b. foo(n) = 5*n
    c. foo(n) = n^5
    d. foo(n) = 5^n

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*
    c. 2D arrays are twice as big as 1D arrays

None

20. Which are all valid identifiers:
    a. Change, 7eleven, m77, foo_bar
    b. float, a, sIlLy, data?
    c. main, seven11, a$, test_case, java

C - 7eleven and
data? are invalid

Section II: 9 questions

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

a) Given **ints** *a*, *b*, and *c*, add *a* and *b* and store the result in *c*.

c = a + b;

b) Given a **double** called *time*, print out "Open" or "Closed" depending on whether or not *time* is greater than or equal to 800 and less than 1700.

```
if(time >= 800 && time < 1700)
    System.out.print("Open");
else
    System.out.print("Closed");
```

c) Given an **char** *c*, print out "1", "2", or "3" depending on the value of *c* ('a', 'b', or 'c'). Use a **switch** statement for your solution.

```
switch(c) {
    case 'a': System.out.print("1"); break;
    case 'b': System.out.print("2"); break;
    case 'c': System.out.print("3"); break;
}
```

4

2. [10 points] Write a method called *divisors*() that takes as a parameter an **int** k, and uses a **for** loop to find and print out the integer divisors of k (i.e. all integers that evenly divide into k). Do not use any other methods - compute the divisors yourself. *divisors*() should return the number of integer divisors of k.

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

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

3. [10 points] One way to calculate square root of a number *a* is to guess that the root is some number *n* between 0 and *a*, and then repeatedly refine your guess *n* by setting n equal to the average of *n* and *a* / *n*, until the difference between your n^2 and *a* is very small. Show how you could implement sqrt() using this method to return the square root to within .00001 of a number passed as a parameter.

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

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

    return n;
}
```

4. [10 points] Write a main() method that reads in four integers and prints out "increasing" if the numbers were entered in increasing order and prints out "not increasing" 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("Increasing");
    else
        System.out.print("Not Increasing");
}
```

5. [10 points] Write a method that tests whether the formula $x^2 - y^2 = z^2$ is true for three integers passed as parameters and returns the result of this test.

```
public static boolean testPythagorean(int x, int y, int z) {
    return x*x - y*y == z*z;
}
```

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) = 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 Coord 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 Coords and returns a new coordinate containing the sum of the other two.
c) A static method called distance() that returns the distance between two Coords. 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 Coords X and Y and adds them and stores the result in Z.

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

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

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

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

Test code:
Coord X = (1,1);
Coord Y = (2,2);
Coord Z = X.add(Y);
```

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 queue, 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 incremememted. 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

```java
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;
    }
}
```

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. 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   B
   c. 3
   d. both 2 and 3
   e. nothing

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

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

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

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

6. 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");
```

   B - the test is true, so false is printed out

   a. true
   b. false
   c. nothing

7. 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

8. Method overloading is when you write a method   *C*
   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 class

9. What is the value of the expression

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

   None - it print out 6 (= 3 + 20 - 20 + 3)

   a. 0
   b. 12
   c. 3

10. 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');
}
```

   D - no break statements

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

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

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

2

13. Public vs. Private:
    a. Public methods can only access public data elements
    b. Private data elements are accessible by methods in the same package *C*
    c. By default, all data elements are private

14. 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

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

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

16. 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   *None*
   c. 2 3 7 42

17. Which is true?
    a. return(a,b); can be used to return two values
    b. primitive types are call-by-reference   *None*
    c. the scope of an object is the class that defines its type

18. 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 nth fibonacci number   *None*
   b. foo(n) = 5*n
   c. foo(n) = n^5
   d. foo(n) = 5^n

3

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*
    c. 2D arrays are twice as big as 1D arrays

None

20. Which are all valid identifiers:
    a. Change, 7eleven, m77, foo_bar
    b. float, a, sIlLy, data?
    c. main, seven11, a$, test_case, java

C - 7eleven and data? are invalid

Section II: 9 questions

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

a) Given **ints** *a*, *b*, and *c*, add *a* and *b* and store the result in *c*.

```
c = a + b;
```

b) Given a **double** called *time*, print out "Open" or "Closed" depending on whether or not *time* is greater than or equal to 800 and less than 1700.

```
if(time >= 800 && time < 1700)
    System.out.print("Open");
else
    System.out.print("Closed");
```

c) Given an **char** *c*, print out "1", "2", or "3" depending on the value of *c* ('a', 'b', or 'c'). Use a **switch** statement for your solution.

```
switch(c) {
    case 'a': System.out.print("1"); break;
    case 'b': System.out.print("2"); break;
    case 'c': System.out.print("3"); break;
}
```

2. [10 points] Write a method called *divisors*() that takes as a parameter an **int** *k*, and uses a **for** loop to find and print out the integer divisors of *k* (i.e. all integers that evenly divide into *k*). Do not use any other methods - compute the divisors yourself. *divisors*() should return the number of integer divisors of *k*.

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

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

3. [10 points] One way to calculate square root of a number *a* is to guess that the root is some number *n* between 0 and *a*, and then repeatedly refine your guess *n* by setting n equal to the average of *n* and *a* / *n*, until the difference between your n^2 and *a* is very small. Show how you could implement sqrt() using this method to return the square root to within .00001 of a number passed as a parameter.

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

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

    return n;
}
```

4. [10 points] Write a main() method that reads in four integers and prints out "increasing" if the numbers were entered in increasing order and prints out "not increasing" otherwise.

```java
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("Increasing");
    else
        System.out.print("Not Increasing");
}
```

5. [10 points] Write a method that tests whether the formula $x^2 - y^2 = z^2$ is true for three integers passed as parameters and returns the result of this test.

```java
public static boolean testPythagorean(int x, int y, int z) {
    return x*x - y*y == z*z;
}
```

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:

- $rec(0) = 73$
- $rec(n) = n * rec(n-1) + rec(n-2)$

```
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 Coord 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 Coords and returns a new coordinate containing the sum of the other two.
c) A static method called distance() that returns the distance between two Coords. 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 Coords X and Y and adds them and stores the result in Z.

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

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

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

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

Test code:
Coord X = (1,1);
Coord Y = (2,2);
Coord Z = X.add(Y);
```

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 queue, 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;
    }
}
```