Thomas Jefferson Invitational Open in Informatics 2012

Exam in Java

Contest Part I (Theoretical—Short Answer)

Do not open until told to do so.

Rules and Instructions:

1. You will have 60 minutes to complete this section. This may not be enough time to complete the entire contest. Do not be alarmed if you do not finish every problem.
2. For each problem, write your answer clearly, neatly, and legibly in the space provided in the answer document.
3. For code analysis problems, assume that all necessary headers and libraries have been included, and that there are no compile or run-time errors.
4. Each problem will be weighted differently, for a total of 50 points. Partial credit may be awarded for partially-correct answers.
5. Work on these problems within your team.
6. Please direct all non-content-related questions to the proctors.
7. Good luck, and have fun!

Short Answer Problem 0

(2 points)

Pascal's triangle is a number array that is often used in combinatorics. See the diagram below for more details. The first row of Pascal's triangle is row 0. The leftmost column in each row is column 0. Thus, row 0 contains 1 column (column 0), row 1 contains column 0 and column 1, and so on. For example, the number in row 5 and column 3 is 10 (the second 10 in the row). The entries of Pascal's triangle are generated using the following procedure:

- If the cell is the leftmost or the rightmost in its column, then its value is 1.
- Else, it is the sum of the two numbers diagonal to it in the previous row (see diagram).

Pascal's Triangle

row 0:					1					
row 1:				1		1				
row 2:			1		2		1			
row 3:		1		3		3		1		
row 4:	1		4		6		4		1	
row 5:	1		5		10		10		5	1

The method defined below is supposed to print the Pascal's triangle value in row 9 and column 5. Unfortunately, one of the lines (marked with an asterisk) may be coded incorrectly. Find the correct code.

```
public static int pascal(int row, int col){
    if(col == 0 || col == row)
        return 1;
    else
        return pascal(row - 1, col - 1) + pascal(row - 1, col + 1); // *
}
public static void main(String[] args){
    System.out.println(pascal(9, 5));
}
```

Replace the line marked by an asterisk with:

A) `return pascal(row - 1, col) + pascal(row + 1, col);`

B) `return pascal(row - 1, col) * pascal(row - 1, col + 1);`

C) `return pascal(row - 1, col - 1) + pascal(row - 1, col);`

D) `return pascal(row + 1, col - 1) + pascal(row + 1, col);`

E) `return pascal(row - 1, col - 1) + pascal(row - 1, col + 1);` (no change)

Short Answer Problem 1

(3 points)

Albert tried to code a Selection Sort to sort an arbitrary list of integers, but he made a mistake. Find the output of the program below, given that there are no compilation or runtime errors. The answer does not require formatting specific to each language (just write down the numbers in the order they appear in the list).

```
static int[] numbers = {7, 3, 5, 2, 4};

static void selection_sort(int len) {
    for(int i = 0; i < len; i++) {
        int max_index = i;
        for(int j = i; j < len; j++)
            if(numbers[j] > numbers[max_index])
                max_index = j;
        numbers[i] = numbers[max_index];
    }
}

public static void main(String[] args) {
    selection_sort(5);
    System.out.println(numbers[0] + " " + numbers[1] + " " + numbers[2]
        + " " + numbers[3] + " " + numbers[4]);
}
```

Short Answer Problem 2

(3 points)

Bessie the cow is throwing a party for all her friends Alex (A), Brian (B), Chloe (C), and Doug (D). However, she accidentally lost all of the RSVP's.

Using Boolean algebra and substituting Alex for A, Brian for B, Chloe for C, and Doug for D, who can come to the party? Each of A, B, C, and D represents a Boolean value that is 1 (true) if the person is coming and 0 (false) if the person is not coming.

Bessie creates the following equation to help you solve for the boolean values A, B, C, and D. All binary operations are done bitwise. For example, $01001\,\text{OR}\,11100 = 11101$. Help her figure out who can come to the party by solving the following.

$$(\text{NOT}(01111)\,\text{OR}(\text{NOT}(01001)\,\text{AND}\,01011)) = AB010\,\text{AND}\,C10D1$$

Short Answer Problem 3

(4 points)

Given the following code segments, determine all output exactly. You may assume that all necessary headers and libraries have been included, and that there are no compile or run-time errors.

a)
```
public static int function(int n){
    int a = 1,i = 1;
    for(i = 1;i <= n + 1;i++){
        int b = 0,j = 1;
        for(j = 1;j <= i + 1;j++)
            b = b + a;
        a = b;
    }
    return a;
}
public static void main(String[] args){
    System.out.println(function(3));
}
```

b)
```
public static int function(int a, int b){
    if(a == 0)
        return 1;
    if(b == 0)
        return 2;
    if(a > 5)
        return -1;
    return function(a+b, Math.abs(a-b)) + function(a+b, Math.abs(a-b)-1);
}
public static void main(String[] args){
    System.out.println(function(2, 1));
}
```

Short Answer Problem 4

(3 points)

The following program should output the value of the bth factorial of a: $a \cdot (a-b) \cdot (a-2b) \cdot \cdots \cdot (a-kb)$, where $(a-kb)$ is positive and $(a-(k+1)b)$ is negative. Unfortunately, one of the lines (marked with an asterisk) may be coded incorrectly. Find the correct code.

```
public static int buggy(int a, int b){
    int c = b;
    int x = a;
    while(a - c > 0)
    {
        a = a * (a-c); // *
        c = c + b;
    }
    return x;
}
```

Replace the line marked by an asterisk with:

A) `x = a * (a - c);`

B) `a = x * (x - c);`

C) `x = x * (a - c);`

D) `x = x * (x - a);`

E) `a = a * (a - c);` (no change)

Short Answer Problem 5

(4 points)

Find all solutions where the following expressions evaluate to TRUE. Express your answers as ordered tuples.
Note: $A + B$ means A OR B, $A \bullet B$ means A AND B, $\overline{A}$ means NOT A.

a) $A + B \bullet C$

b) $A + \overline{B \bullet C}$

c) $\overline{\overline{(A + B) \bullet C} \bullet ((A \bullet C) + C) \bullet (B \bullet C + A)} + \overline{(A \bullet C) + (A \bullet B)}$

Short Answer Problem 6

(4 points)

Normally, when adding two non-decimal numbers, we convert the number to decimal, then add, then convert back to the particular base from which we started. For example,

$$101_2 + 110_2 = 5_{10} + 6_{10} = 11_{10} = 1011_2$$

However, sometimes it's faster to do arithmetic in that particular base. Consider, for example, addition in base 2. Addition in binary is very similar to addition in decimal. The only difference is what causes carries. $0 + 0 = 0$, $0 + 1 = 1 + 0 = 1$, and $1 + 1 = 0$, carrying a 1 to the next column. Look at the following example:

```
                        1                   1
  1001011             1001011             1001011
+   10010    =>     +   10010    =>     +   10010
---------           ---------           ---------
        1                  01             1011101
```

Thus, the answer is 1011101_2. Evaluate each of the following expressions. Leave your answer in the given base.

a) $1001_2 + 110111_2 =$

b) $100101001110 01_2 + 1010100111_2 =$

c) $A3C01FF_{16} + 29DD92C1B_{16} =$

d) $10010101_2 - 1100111_2 =$

e) $427_8 \cdot 31_8 =$

f) $\frac{10101000_2}{100_2} =$

g) $FFFFF_{16}^3 + 3 \cdot FFFFF_{16}^2 + 3 \cdot FFFFF_{16} + 1 =$

Short Answer Problem 7

(4 points)

Pickle and Rhonun play a game. In this game each picks a random base 10 number. They then add up all the digits and that becomes the new base that they must convert to. They then add up all the digits (letters represent their corresponding value: ex. $A \rightarrow 10$) of this base and this becomes base 10. Whoever has the higher number wins. They play three times.

1st game:
Pickle: 12_{10}
Rhonun: 11_{10}

Explanation: Because Rhonun picked 11, he must convert to base $(1 + 1) = 2$. 11_{10} is equal to 1011_2. The sum of the digits is 3_{10}. Pickle picked 12_{10}. This converted to base $(1 + 2) = 3$ is 110_3. The sum of the digits is 2_{10}. In this round Rhonun won with a score of 3 and Pickle lost with a score of 2.

2nd game:
Pickle: 161_{10}
Rhonun: 221_{10}

3rd game:
Pickle: 734_{10}
Rhonun: 803_{10}

a) Who won the overall game (ie. won the most individual times)?

b) Who had the highest total score and what was it?

Short Answer Problem 8

(4 points)

Given the following code segments, determine all output exactly. You may assume that all necessary headers and libraries have been included, and that there are no compile or run-time errors. There will be points awarded for each line of the input, so partial credit will be awarded.

```
public static int function(int n){
    int[] mystery = new int[n + 1];
    for(int i = 0; i <= n; i++)
        mystery[i] = i;
    for(int i = 2; i <= n; i++)
        if(mystery[i] == i)
            for(int j = 2 * i; j <= n; j += i)
                mystery[j] += i;
    return mystery[n];
}
public static void main(String[] args){
    System.out.println(function(12));
    System.out.println(function(16));
    System.out.println(function(39));
    System.out.println(function(99));
    System.out.println(function(100000000));
}
```

Short Answer Problem 9

(4 points)

Albert is in line to order some ice cream at the ice cream store. The line is a strange one and is controlled by the cashier. The line's properties can take two form: S (stack) and Q (queue). In S form, whoever was the most recent person to enter the line is the next to get ice cream. In Q form, whoever has been in the line the longest is the next to get ice cream. **Initially, the line has form** Q.

Over the course of the day, there are several events. If the event is of type 1, a new person enters the line. Each person has a name that is a string of lowercase letters with length at most 100. If the event is of type 2, the next person in line gets ice cream, where the "next" person is determined by the form of the line. If the event is of type 3, then the line switches form, from Q to S or vice versa.

Given the events, determine the order in which people get ice cream. It is not guaranteed that everyone gets ice cream.

As an example, consider this list of events:

```
9
1 Albert
1 Lbert
1 Bert
3
1 Ert
2
2
3
2
```

The correct solution for this input would be:

```
Ert
Bert
Albert
```

Find the output for the following case (the list runs top to down, left to right):

```
27          1 E         2           3           3
1 A         1 F         3           1 J         2
1 B         1 G         2           1 K         3
1 C         1 H         2           2           2
1 D         3           1 I         3
3           2           2           2
```

Short Answer Problem 10

(5 points)

Mathematical expressions can be evaluated in many ways; the three most common are prefix, infix, and postfix [1] . In prefix, the operator comes before the operands (the numbers being operated on); in infix, the operator comes between the operands (our "normal" mathematical evaluation form); and in postfix, the operator comes after the operands.

This means that, for a given expression, there are three equivalent ways to express that expression. Since these concepts are best taught through examples, examine the examples below and answer the questions that follow. Note that ↑ is the exponential operator, and that `/` is the division operator, and that the standard order of operations is still applicable. Also note that the order of numbers for all three forms must be the same.

i) Infix: `1 + 2 + 3`
Prefix: `+ + 1 2 3`
Postfix: `1 2 + 3 +`

ii) Infix: `1 * 2 + 3`
Prefix: `+ * 1 2 3`
Postfix: `1 2 * 3 +`

iii) Infix: `1 * ( 2 + 3 )`
Prefix: `* 1 + 2 3`
Postfix: `1 2 3 + *`

iv) Infix: `1 * 2 + 3 * 4`
Prefix: `+ * 1 2 * 3 4`
Postfix: `1 2 * 3 4 * +`

v) Infix: `1 + 2 / 3 ↑ 4`
Prefix: `+ 1 / 2 ↑ 3 4`
Postfix: `1 2 3 4 ↑ / +`

vi) Infix: `((1 + 2) / 3) ↑ 4`
Prefix: `↑ / + 1 2 3 4`
Postfix: `1 2 + 3 / 4 ↑`

For each of the problems below, write the expression given in either prefix, infix, or postfix in the other two -fixes.

a) Infix: `1 + 3 + 2 + 8`

b) Infix: `(1 + 3)/(2 + 8)`

c) Prefix: `+ - - 1 4 6 7`

d) Postfix: `1 4 9 - + 2 3 * ↑`

e) Infix: `1 + 3 / 2 + 8`

f) Infix: `1 ↑ 2 * 3 + 4 - 5`

g) Prefix: `/ + * 1 3 5 7`

h) Postfix: `1 3 * 5 7 - ↑`

i) Infix: `1 - 2 + 3 * 4 ↑ 5`

j) Infix: `1 - 4 ↑ 3 / 5`

k) Prefix: `/ * ↑ 1 * 2 3 4 5`

l) Postfix: `3 4 7 / 5 * ↑`

For these last two problems, identify whether the expression given is in prefix, infix, or postfix, and then evaluate the expression.

m) `1 2 + 3 4 + * 6 5 - ↑ 6 8 + /`

n) `↑ * * + / 1 2 / 3 2 5 4 - 7 8`

[1] Although this can be extended to more than just mathematical expressions, we will use only mathematical expressions in the questions below.

Short Answer Problem 11

(5 points)

Suppose we have a gigantic complete binary tree with an infinite height (an infinite number of layers). The first row is row 0 and row numbers increase as we go further from the root of the binary tree. The leftmost node in each row is marked as column 0, and column numbers increase going right. Row 0 has 1 node, row 1 has 2 nodes, and so on (with row x having 2^x nodes).

For each of the following problems, you are given two distinct nodes on a binary tree. Find the length of the shortest path between the two nodes. Hint: all paths through a binary tree have a similar pattern. For the larger inputs, it might be helpful to consider the binary representation of the column number.

a) First node: row 2 column 0
   Second node: row 2 column 1

b) First node: row 2 column 0
   Second node: row 2 column 3

c) First node: row 3 column 0
   Second node: row 4 column 12

d) First node: row 8 column 49
   Second node: row 8 column 48

e) First node: row 100 column 3
   Second node: row 102 column 102

Short Answer Problem 12

(5 points)

When a computer computes the value of exponentation (a^b), the most common way is to iteratively multiply a by itself b times. Unfortunately, this method is fairly slow and there exists a much faster way. This way is often referred to as "power of two" exponentiation because powers of 2 are involved. Suppose we know the values of a^1, a^2, a^4, a^8, and so on. With only power of 2 powers of a, it is possible to calculate a^b for any b. For example, we can calculate a^5 by noting that $a^5 = a^1 \cdot a^4$. As another example, $a^{15} = a^1 \cdot a^2 \cdot a^4 \cdot a^8$. Using this idea, we have an implementation of "power of 2" exponentiation below. Unfortunately, it has some errors. (Note: $b \geq 0$ and $0^0 = 1$).

```
public static int pow(int a, int b){
    if (b <= 0){
        return 1;
    }
    else if (b == 1){
        return a;
    }
    else{
        int tmp = pow(a, b / 2);
        return tmp * tmp;
    }
}
```

Answer the following:

a) What is the error in the method? (Hint: it might help to try some cases and see if you can find a pattern.)

b) Give an example of a pair of integers a and $b > 1000$ such that pow(a, b) is calculated *correctly*.

c) Modify or add one or more lines in such a way that pow(a, b) gives the correct result for all inputs (do this on the code in the answer document).

d) What is the runtime complexity of "power of 2" exponentiation? Give your answer in Big-O notation.