

CS 14: Data Structures and Algorithms

Monday, October 13, 2003

Quiz 3, Form:

Name: \_\_\_\_\_

Login: \_\_\_\_\_

ID Number: \_\_\_\_\_

Signature: \_\_\_\_\_

This quiz is worth **54** points and lasts **12** minutes. You may use **ONLY** the test's last sheet (front and back pages) for **scratch** work, and your final answers must be **BRIEF, CLEAN, COHERENT, LEGIBLE**, and written **ONLY** in the boxed spaces provided. ■

Good luck!

1. [5 points] (×3) Write a **simple** C++ implementation for the function **int sumOfPos(const int a[ ], const unsigned int length)** which takes as arguments an array of integers, **a**, and its length, **length**, and returns the **sum** of **only** those integers stored in the array which are **positive**.

2. [2 points] (×3) In the previous question, what is the effect of the keyword **constant** written before the array argument? Please explain your answer with a complete sentence.

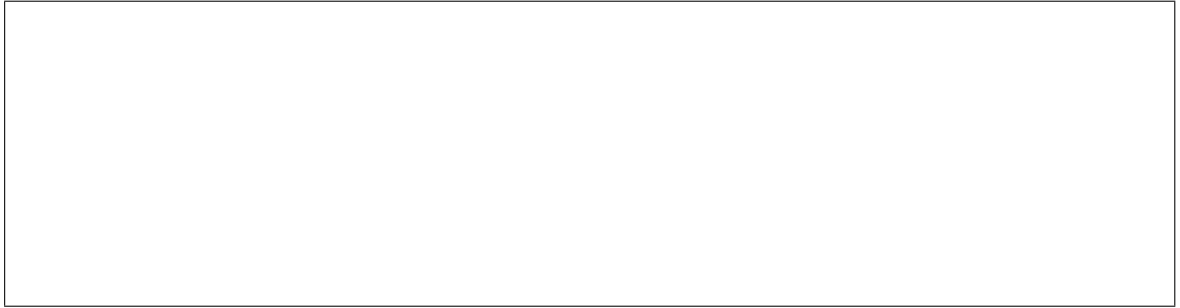
3. [7 points] ( $\times 3$ ) The binomial coefficient  $\text{binCoeff}(n, p)$  is the number of distinct ways you can choose  $p$  items out of a collection of  $n$  items. For example, the number of ways you can choose 2 apples from a basket of 10 apples is  $\text{binCoeff}(10, 2)$  and is equal to 45. As it turns out, the binomial coefficient  $\text{binCoeff}(n, p)$  can be defined recursively by the following equations:

$$\begin{aligned}\text{binCoeff}(n, p) &= \text{binCoeff}(n-1, p) + \text{binCoeff}(n-1, p-1), & \text{for } n \geq p \geq 1, \\ \text{binCoeff}(n, p) &= 1, & \text{if } n = p, \\ \text{binCoeff}(n, 0) &= 1, & \text{for } n \geq 0.\end{aligned}$$

The first line is the actual recursive relation, whereas the remaining lines define the base cases. Note that the first argument must never be smaller than the second. In the box below write a recursive function `int recBinCoeff(unsigned int n, unsigned int p)` which computes the binomial coefficient defined above. Hint: this question is not as hard as it looks; it can be solved in as little as 3 lines, not counting the line for the function header. If you need to deal with invalid inputs, use an exception.

4. [5 points] Suppose you have three variables,  $x$ ,  $y$ , and  $z$ , each a **pointer** to a node in a doubly linked list. In fact, they're linked like this:  $x \rightleftharpoons y \rightleftharpoons z$ . Assuming that you have direct access to the member variables of the `Node` class, write the C++ code required to delete the node pointed to by  $y$ . You don't need to write a function, only the code fragment that deletes  $y$ . You may assume that none of the pointers are null.

5. [7 points] Now solve the previous question again, but when the **only** variable you know of is  $y$ . In other words, you have the **same list** again, but now you only have  $y$  and you want to delete the node it points to. You may assume that  $y$  is not null.



ANY WORK BELOW THIS LINE OR ON THE NEXT PAGE(S) WILL BE CONSIDERED SCRATCH WORK AND WILL **NOT** BE GRADED. **NO** EXCEPTIONS WILL BE MADE.

# Answer Key for Exam A

This quiz is worth **54 points** and lasts **12 minutes**. You may use **ONLY** the test's last sheet (front and back pages) for **scratch work**, and your final answers must be **BRIEF, CLEAN, COHERENT, LEGIBLE**, and written **ONLY** in the boxed spaces provided. ■

Good luck!

1. [5 points] (×3) Write a **simple** C++ implementation for the function **int sumOfPos(const int a[], const unsigned int length)** which takes as arguments an array of integers, **a**, and its length, **length**, and returns the **sum** of **only** those integers stored in the array which are **positive**.

**Answer:**

```
int sumOfPos(const int a[ ], const unsigned int length)
{
    int sum = 0;
    for (unsigned int i = 0; i < length; ++i)
    {
        if (a[i] > 0)
            { sum += a[i]; }
    }
    return sum;
}
```

2. [2 points] (×3) In the previous question, what is the effect of the keyword **constant** written before the array argument? Please explain your answer with a complete sentence.

**Answer:** The keyword **constant** tells the compiler to enforce that whatever appears after that keyword will remain unchanged during the execution of its enclosing block. Thus, in this case, it says that the function in question is not allowed to change the elements of the array while the function is executing. This is useful to users of the function because they will know that the array will not be modified by this function, without ever having to see the function's implementation.

3. [7 points] (×3) The binomial coefficient  $\text{binCoeff}(n, p)$  is the number of distinct ways you can choose  $p$  items out of a collection of  $n$  items. For example, the number of ways you can choose 2 apples from a basket of 10 apples is  $\text{binCoeff}(10, 2)$  and is equal to 45. As it

turns out, the binomial coefficient `binCoeff(n, p)` can be defined recursively by the following equations:

$$\begin{aligned} \text{binCoeff}(n, p) &= \text{binCoeff}(n-1, p) + \text{binCoeff}(n-1, p-1), & \text{for } n \geq p \geq 1, \\ \text{binCoeff}(n, p) &= 1, & \text{if } n = p, \\ \text{binCoeff}(n, 0) &= 1, & \text{for } n \geq 0. \end{aligned}$$

The first line is the actual recursive relation, whereas the remaining lines define the base cases. Note that the first argument must never be smaller than the second. In the box below write a recursive function `int recBinCoeff(unsigned int n, unsigned int p)` which computes the binomial coefficient defined above. Hint: this question is not as hard as it looks; it can be solved in as little as 3 lines, not counting the line for the function header. If you need to deal with invalid inputs, use an exception.

**Answer:** Actually, returning an `int` is not a good idea because binomial coefficients grow very quickly out of the range allowed by `ints`. We should return 64-bit integers, called **long long ints**. I didn't mention this in the quiz because I didn't want to un-necessarily confuse people.

```
long long int recBinCoeff(unsigned int n, unsigned int p)
{
    if (n < 0 or p < 0 or n < p) // invalid inputs
    { throw std::domain_error("Invalid inputs."); }
    else
    if (n == p or (n >= 0 and p == 0)) // base cases
        return 1;
    else // recursive calls
        return ( recBinCoeff(n-1, p) + recBinCoeff(n-1, p-1) );
}
```

4. [5 points] Suppose you have three variables, `x`, `y`, and `z`, each a **pointer** to a node in a doubly linked list. In fact, they're linked like this:  $x \rightleftharpoons y \rightleftharpoons z$ . Assuming that you have direct access to the member variables of the `Node` class, write the C++ code required to **delete** the node pointed to by `y`. You don't need to write a function, only the code fragment that deletes `y`. You may assume that none of the pointers are null.

**Answer:**

```
(x -> next) = z;
(z -> previous) = x;
delete y;
```

5. [7 points] Now solve the previous question again, but when the **only** variable you know of is `y`. In other words, you have the **same list** again, but now you only have `y` and you want to delete the node it points to. You may assume that `y` is not null.

**Answer:** The answer is almost exactly the same, and very easy if you've solved the previous question first. Since `x` is `y`'s previous (`x = y->previous`), and `z` is `y`'s next (`z = y->next`), we have:

```
((y -> previous) -> next) = (y -> next);  
((y -> next) -> previous) = (y -> previous);  
delete y;
```