

CS 14: Data Structures and Algorithms

Monday, November 3, 2003

Quiz 5, Form:

Name: _____

Login: _____

ID Number: _____

Signature: _____

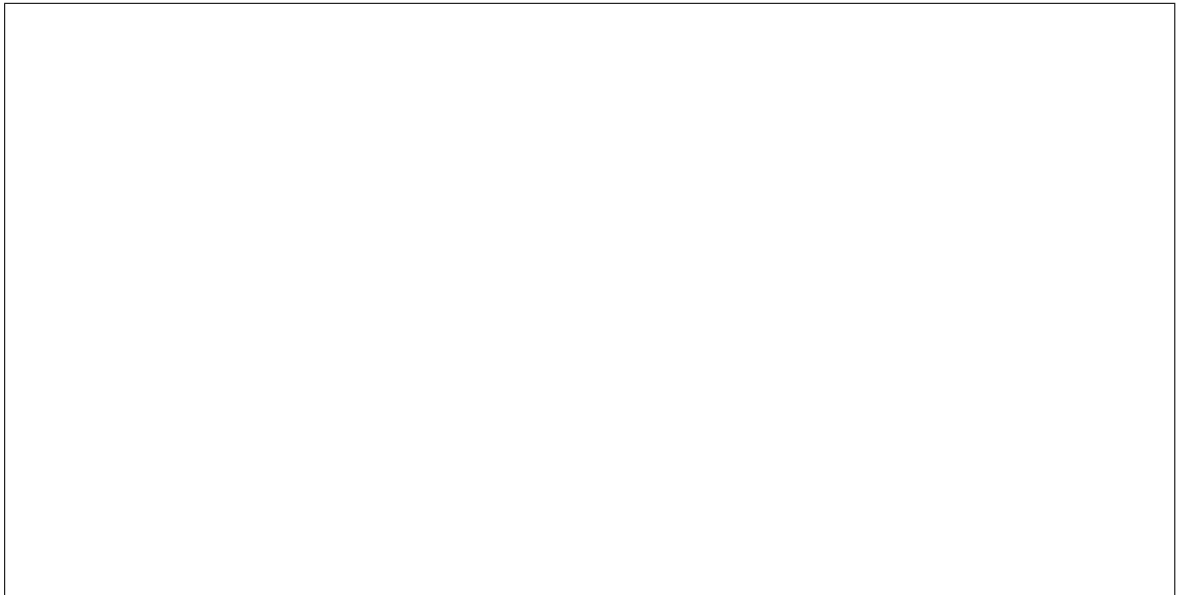
This quiz is worth 10 points and lasts 12 minutes. Good luck!

1. [5 points] **Briefly** explain (1) [1 point] what the following function does and (2) [2 points] why it's **not** the most efficient implementation, then [2 points] write a **simpler** and **more efficient** version of it.

```
bool f(const int a[], const unsigned int length)
{
    bool result = true;
    for (unsigned int i = 0; i < length; ++i) {
        if (a[i+1] < a[i]) { result = false; }
    }

    if (result == true)
        return true;
    else
        return false;
}
```

2. [5 points] Write the C++ code for the recursive implementation of the mergeSort algorithm, but **don't** worry about implementing the actual merge part. In other words, do **only** the first part of what we did in lecture. Here's the function header: `void mergeSort(const int a[], const unsigned int start, const unsigned int end)`.



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 **10 points** and lasts **12 minutes**. Good luck!

1. [5 points] **Briefly** explain (1) [1 point] what the following function does and (2) [2 points] why it's **not** the most efficient implementation, then [2 points] write a **simpler** and **more efficient** version of it.

```
bool f(const int a[], const unsigned int length)
{
    bool result = true;
    for (unsigned int i = 0; i < length; ++i) {
        if (a[i+1] < a[i]) { result = false; }
    }

    if (result == true)
        return true;
    else
        return false;
}
```

Answer: The function returns **true** if the array is in **ascending** order and returns **false** otherwise. It's not the most efficient implementation for **two** reasons: (1) it **always** goes through the **entire** array before returning a result, whereas it's possible in many cases to return early, as soon as we detect that the array is not in ascending order, and (2) the last part is totally redundant; just return the result! Here's the most efficient solution I can think of:

```
bool isAscending(const int a[], const unsigned int length)
{
    for (unsigned int i = 0; i < length; ++i)
    {
        // if the _next_ element is _smaller_ than
        // the _current_ element, then the array
        // is _not_ in ascending order and we can
        // return false immediately, without having
        // to continue the loop.

        if (a[i+1] < a[i])
```

```
        { return false; }
    }

    return true;
}
```

2. [5 points] Write the C++ code for the recursive implementation of the mergeSort algorithm, but **don't** worry about implementing the actual merge part. In other words, do **only** the first part of what we did in lecture. Here's the function header: void mergeSort(const int a[], const unsigned int start, const unsigned int end).

Answer:

```
void mergeSort(const int a[],
               const unsigned int start,
               const unsigned int end)
{
    if (start < end)
    {
        int middle = (start + end) / 2;

        mergeSort(a, start, middle);
        mergeSort(a, middle + 1, end);

        merge(a, start, middle, end);
    }
}
```