Computer Science I
Mid-Term - Sep 19th, 2011

Max Marks: 50 **Time: 3 Hrs**

1. Show that the big-O notation is transitive – i.e. for any three functions on integers f, g and h, $f = O(g)$ and $g = O(h)$ implies $f = O(h)$. **Marks**: 1

2. Sort the following 10 functions in the increasing sequence of their Order. In other words sort them as a sequence $f_1, f_2, f_3, \ldots, f_{10}$ such that $f_i=O(f_{i+1})$ for all $i=1,\ldots,9$. **Marks**: 2

   $10^{-3}n^3$, $n(\log n)^2$, 10^{12}, $10^6(\log n)^{100}$, $n^{\log\log n}$, $\log\log n$, $1/\log n$, $2^{\log\log n}$, $n^{\log\log n}$, $n!$

3. Consider the grammar $S : aS \mid aSbS \mid \in$, representing a set of strings made up of terminal symbols a and b. Show that this set is $\{x \mid \text{any prefix of } x \text{ has at least as many a's as b's}\}$. A prefix of a string is a substring starting from the beginning of the string. **Marks**: 3

4. Consider the grammar shown below with terminal symbols a, b and c. **Marks**: 5

   $S : aScB \mid A \mid b$

   $A : cA \mid c$

   $B : d \mid A$

   Which of the following are valid strings under this grammar, and why?

   a. abcd
   b. aabadbc
   c. aabccccd
   d. ababcdcd
   e. accccd

5. **Insertion Sort on small arrays within Merge Sort**

   Although Merge Sort runs in $O(n \log n)$ worst case time and insertion sort runs in $O(n^2)$ worst case time (remember the shifts to be done for every insertion), the constant factors in insertion sort make it faster for small n. Thus it makes sense to use insertion sort within Merge Sort when the sub-problems become sufficiently small. Consider a modification of Merge Sort in which n/k subsets of length k are sorted individually using insertion sort and then merged using the standard merging mechanism. k is a value to be determined.

   a. Show that the n/k sub-lists, each of length k, can all be sorted by insertion sort in worst case $O(nk)$ time. **Marks:** 2

   b. Show that the sorted sub-lists can be merged together in $O(n \log(n/k))$ time in the worst case. **Marks:** 5

c. Explain the way the overall running time of the algorithm changes as $\mathbf{k}$ varies from small constant values to $\mathbf{n}$. **Marks:** 1

d. What is the optimum value of $\mathbf{k}$ one should choose while ensuring an asymptotic worst case time of $O(n \log n)$. **Marks:** 1

6. Let $\mathbf{X}$ and $\mathbf{Y}$ be two arrays each containing $\mathbf{n}$ numbers already in sorted order. Give an algorithm that runs in worst case $O(\log n)$ time to find the median of all the $2n$ elements in $\mathbf{X}$ and $\mathbf{Y}$ together. **Hint**: Use recursion and binary search. **Marks:** 5

7. Write two versions of **swap** in C++ – both swap a pair of integers located elsewhere – but with slightly different arguments.

   void swap1(int *i, int *j) and **void swap2(int &i, int &j)**

   Also show how these swap functions would be called from say the main program shown below. Complete the argument list for the two calls to swap to show how the two functions would be invoked.

```
int main()
{
        int p = 1, q = 2;
        swap1(...); // to swap p and q

        int x = 10, y = 20;
        swap2(...); // to swap x and y
}
```

Marks: 5

8. Write declarations for the following: **Marks:** 5
   a. a pointer to a character;
   b. an array of 10 objects of type **T**
   c. a constant double
   d. a constant pointer to an array of characters
   e. a reference to an object of type **T**
   f. a constant integer serving as a common flag across all instances of a class
   g. a reference to an array of objects of type **T**
   h. a method called **doSomething** of a class **C** taking arguments of type pointer to character and reference to integer and returning no value.
   i. a method **safeDo** of a class **C** that takes no arguments returns an integer and guarantees that it has no effect on the state of the object on which it is called.
   j. a function **getHandle** that takes a reference to an array of objects of type **T** and another object **x** of type **T** as arguments and returns a reference to the element of the input array that is say closest to **x**, while guaranteeing that the input array is not tampered with during the execution of **getHandle**.

   Also give appropriate initializations for a-g. Only the declarations please for h-j.

9. Rewrite the following **for** statement **Marks: 3+2**

```
for (int i = 0; i<max_length; i++)
{
        if (inputs_line[i]==’?’) quest_count++;
}
```

a. as an equivalent **while** statement.

b. to use a pointer as the loop variable—that is, so that the test is of the form ***p=='?’**.

10. All of the following are illegal names for variables in a C++ program. Explain what is wrong with each. **Marks: 2 ½**

a. 2nd_coord
b. y-val
c. double
d. x:y_ratio
e. purple&orange

11. What will the following program output? **Marks: 5**

```
#include <iostream>
using namespace std;

long toDo(long n, long* flist)
{
    if (n==0) return -1;

    if (n < 0) n = -n;

    if (n = = 1) return 0;

    int i = 0, d = 2;

    while (n>1)
    {
        flist[i] = 0;
       while (n%d == 0 && n > 1)
       {
           if (flist[i] == 0)
           {
               flist[i] = d; i++;
           }
           n /= d;
       }
       d++;
    }
    return i;
}
```

```
int main()
{
        long flist[100];

        for (long n=1; n<=100; n++)
        {
                int k = toDo(n, flist);
                cout << n << “: “;
                for (int i=0; i<k; i++)
                {
                        cout << flist[i] << " ";
                }
                cout << endl;
        }
}
```

12. Find at least 4 compilation errors and 1 error that will surface only at runtime (assuming you have removed all the compilation errors) in the following piece of code. There are in fact many more errors in this piece of code – bonus for finding more than 1 runtime and 4 compilation errors. **Marks: 2 ½**

```
#include <iostream>
using namespace std;

class C
{
    public:
        C(void) : x(0), y(NULL) {}                         //(1)
        ~C(void) { delete y; }                             //(2)
        void doThis(void) const { doSomething(); }         //(3)
        int doThat(C* c) { x = c.getX(); }                 //(4)
        int& getX(void) { return x; }                      //(5)

    private:
        int x, *y;                                         //(6)
        void doSomething(void) { x = (x > *y)? x : y; }    //(7)
}

int foo(void)
{
    C  *c = new C;                                         //(8)
    *c->getX();                                            //(9)
    delete c;                                              //(10)
}

int main()
{
    C a, b;                                                //(11)
    a.doThis;                                              //(12)
    b.doThat(a);                                           //(13)
    b.doSomething();                                       //(14)
    foo();                                                 //(15)
    cout << a << " " << b.x;                               //(16)
}
```

The line numbers are shown as comments on the right. Use these to refer to the lines where you think there is an error and explain what the error is.