Computer Science 7A – Final Exam – Fall ’12 Name_____

Show your work for credit. Write all responses on separate paper. Do not use a computer.

1. List all the key words in C++ that you can remember. Note: these are not library words like, “cout.”

2. Find all errors in the complete program below and classify them as either a syntax error, run-time errors or a logical error. If there are no errors, what is the output of the code?

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

int main()
{
   int n = 2000;; // surprisingly, the extra semicolon is just ignored.
   cout << n << endl;
   n *= n;
   cout << n << endl;
   n *= n;
   cout << n << endl;
}
```

3. Find all errors in the complete program below and classify them as either a syntax error, run-time errors or a logical error. If there are no errors, what is the output of the code?

```
#include <iostream>
#include <cmath>
using namespace std;
// Compute the solutions to 3x^2 – 3x -6 = 0
int main()
{
   double a = 3.0;
   double b = -3.0;
   double c = -6.0;
   double d = b*b - 4.0*a*c;
   double x1 = (-b + sqrt(d))/2.0*a;
   double x2 = (-b - sqrt(d))/2.0*a;
   cout << x1 << '\t' << x2 << endl;
}
```

4. Find all errors in the complete program below and classify them as either a syntax error, run-time errors or a logical error. If there are no errors, what is the output of the code?

```
#include <iostream>
#include <cmath>
using namespace std;
// Compute the solutions to x^2 – 10000000000x + 1 = 0
int main()
{
   double a = 1e0;      // == 1.0
   double b = -1e10;    // == -10,000,000,000.0
   double c = 1e0;      // == 1.0
   double d = b*b - 4.0*a*c;
   double x1 = (-b + sqrt(d))/(2.0*a);
   double x2 = (-b - sqrt(d))/(2.0*a);
   cout << x1 << '\t' << x2 << endl;
}
```

5. Consider the following (almost) complete program:

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

// (a) prototypes here

int main()
{
   const int size = 6;
   int a[size] = {5, 2, 3, 1, 13, 8};
   show(a,size);
   mystery(a,size);
   show(a,size);
}

void show(int a[], int n)
{
   // (b) define show
}

void mystery(int a[], int n)
{
   int temp;
   for (int i=0; i<n/2; i++)
   {
         temp     = a[i];
         a[i]     = a[n-i-1];
         a[n-i-1] = temp;
   }
}
```

a. Write prototypes for the functions show() and mystery().
b. Write a function definition for show() that will print out the contents of the array separated by spaces.
c. Given that an **`int`** requires 4 bytes of memory for storage, how much memory is required by the declaration of the array?
   `int a[size] = {5, 2, 3, 1, 13, 8};`
d. If the hexadecimal value of `&a` is `0044FB44`, what is the value of `&a[5]` ?

6. Write a program that takes a 6-digit integer (i.e., in the range 100,000 to 999,999) and then constructs and outputs the integer whose digits are in the reverse order of the input. For example, if 247315 is input, then 513742 is output.

7. Write the output of the program below:

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

int main()
{
   double x = 3.0;
   int n = 5;
   double y = 1.0, z = x;
   int i = n;
   cout << "z = " << z << endl;
   while (i > 0) // INVARIANT: y*pow(z,i) == pow(x,n)
         if (i%2 == 0) {
               z *= z;
               i /= 2;
               cout << "z = " << z << endl;
         }
         else {
               y *= z;
               --i;
               cout << "y = " << y << endl;
         }
         cout << x << "^" << n << " = " << y << endl;
}
```

8. What's going on in the code below? Explain. Note that the output of the program is shown below.

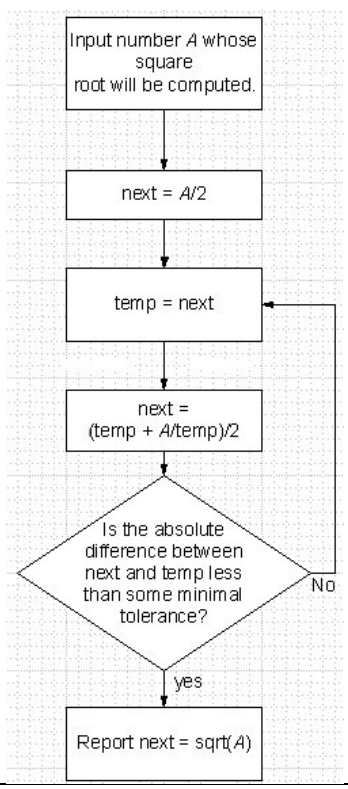
```
#include <iostream>
#include <climits>
using namespace std;

int main()
{
   unsigned long long c = 0;
   for (int i = 0; i < sizeof(unsigned long long) * 8; ++i) {
         unsigned long long d = 1;
         for(int j = 0; j < i; ++j) d *= 2;
         c += d;  //c |= 1 << i;
   }
   cout << c << endl << c+1 << endl;
   unsigned long long f = ULLONG_MAX;
   cout << "\nf = " << f << endl << f+1 << endl;
}
```

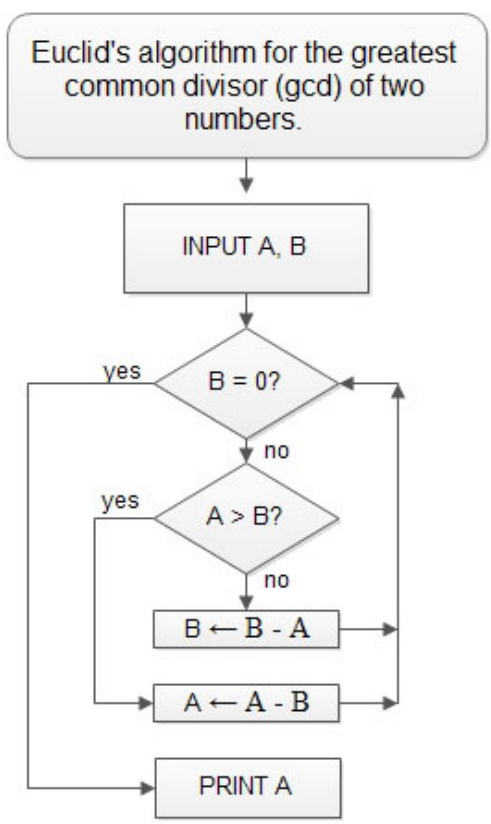
Here's the output:

```
18446744073709551615
0

f = 18446744073709551615
0
Press any key to continue . . .
```

9. Write a C++ program to implement the algorithm described by the flow chart at right.

10. Write a C++ program to implement the algorithm described by the flow chart below.





11. Suppose the file mystery.txt contains the text below:

The whole thing starts about twelve, fourteen or seventeen.

What is the output of the program listed below:

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

int main()
{
	string word;
	int i = 0;
	ifstream read("mystery.txt");
	while(read >> word)
	{
		if(i < word.length())
			cout << word[i];
		++i;
	}
	cout << endl;
}
```

12. Describe the three things you learned in this course that you think are most important.

Computer Science 7A – Final Exam Solutions – Fall ’12

1. List all the key words in C++ that you can remember. Note: these are not library words like, “cout.”
   SOLN: From http://en.cppreference.com/w/cpp/keyword
   They say, “This is a list of reserved keywords in C++. Since they are used by the language, these keywords are not available for re-definition or overloading.” The words we used are in black/bold.

alignas (since C++11)	**enum**	**return**
alignof (since C++11)	explicit	**short**
and	export	**signed**
and_eq	extern	**sizeof**
asm	**false**	static
auto(1)	**float**	static_assert(since C++11)
bitand	**for**	static_cast
bitor	friend	struct
bool	**goto**	**switch**
break	**if**	template
case	inline	this
catch	**int**	thread_local(since C++11)
char	**long**	throw
char16_t(since C++11)	mutable	**true**
char32_t(since C++11)	**namespace**	try
class	**new**	typedef
compl	noexcept(since C++11)	typeid
const	not	typename
constexpr(since C++11)	not_eq	union
const_cast	nullptr (since C++11)	**unsigned**
continue	operator	**using**(1)
decltype(since C++11)	or	virtual
default(1)	or_eq	**void**
delete(1)	private	volatile
do	protected	wchar_t
double	public	**while**
dynamic_cast	register	xor
else	reinterpret_cast	xor_eq

2. Find all errors in the complete program below and classify them as either a syntax error, run-time errors or a logical error. If there are no errors, what is the output of the code?

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

int main()
{
    int n = 2000;;
    cout << n << endl;
    n *= n;
    cout << n << endl;
    n *= n;
    cout << n << endl;
}
```

```
SOLN: If there were no limitations on the
size of an int,the program would output

2000
4000000
16000000000000

But modern compilers usually have INT_MAX
at 2^31 = 2147483648, so the last value
would not compute properly.  This is a
run-time error.
```

3. Find all errors in the complete program below and classify them as either a syntax error, run-time errors or a logical error. If there are no errors, what is the output of the code?

```
#include <iostream>
#include <cmath>
using namespace std;
// Compute the solutions to 3x^2 – 3x -6 = 0
int main()
{
    double a = 3.0;
    double b = -3.0;
    double c = -6.0;
    double d = b*b - 4.0*a*c;
    double x1 = (-b + sqrt(d))/2.0*a;
    double x2 = (-b - sqrt(d))/2.0*a;
    cout << x1 << '\t' << x2 << endl;
}
```

```
SOLN: There is a logic error here.
According the precedence of operators,
multiplication and division are “left
to right” associative.

See http://en.cppreference.com/w/cpp/language/operator_precedence
for a fairly complete listing of C++
operator precedence.

Parentheses are required then to
compute (2.0*a) before dividing.
(-b + sqrt(d))/(2.0*a)
```

4. Find all errors in the complete program below and classify them as either a syntax error, run-time errors or a logical error. If there are no errors, what is the output of the code?

```
#include <iostream>
#include <cmath>
using namespace std;
// Compute the solutions to x^2 – 1e10x + 1 = 0
int main()
{
      double a = 1e0;      // == 1.0
      double b = -1e10;    // == -10,000,000,000.0
      double c = 1e0;      // == 1.0
      double d = b*b - 4.0*a*c;
      double x1 = (-b + sqrt(d))/(2.0*a);
      double x2 = (-b - sqrt(d))/(2.0*a);
      cout << x1 << '\t' << x2 << endl;
}
```

SOLN: The formula used here is computationally unstable. That is, since an 8 byte value for a double (see http://en.cppreference.com/w/cpp/language/types) ranges between the minimum value of $\pm 2.225,073,858,507,201,4 \cdot 10^{-308}$ to the max of $\pm 1.797,693,134,862,315,7 \cdot 10^{308}$, the mantissa is less than 20 digits, the value of the discriminant, d = 1e20 – 4, will be truncated to = 1e20, whose square root, when added to to b will give 0 instead of
(-1e10 + sqrt(99999999999999999996))/2
~(-1e10 + 9.999999999999999998e9)/2
~-9.999999999999999948871108451 17e-11

5. Consider the following (almost) complete program:

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

// (a) prototypes here

int main()
{
    const int size = 6;
    int a[size] = {5, 2, 3, 1, 13, 8};
    show(a,size);
    mystery(a,size);
    show(a,size);
}

void show(int a[], int n)
{
    // (b) define show
}

void mystery(int a[], int n)
{
    int temp;
   for (int i=0; i<n/2; i++)
    {
            temp     = a[i];
            a[i]     = a[n-i-1];
            a[n-i-1] = temp;
    }
}
```

SOLNs:

a. Write prototypes for the functions `show()` and `mystery()`.
SOLN:
```
void show(int [], int)
void mystery(int [], int)
```

b. Write a function definition for show() that will print out the contents of the array separated by spaces. SOLN:
```
void show(int a[], int n)
{
    for(int i = 0; i < size; ++i)
    {
        cout << a[i] << ' ';
    }
    cout << endl;
}
```

c. Since there a 6 integers, and each one requires 4 bytes, the declaration requires 24 bytes.

d. a[5] will be an addition 20 bytes further along in memory. In hexadecimal, 20 = 14, &a[5] = 0044FB44 + 14 = 0044FB58

6. Write a program that takes a 6-digit integer (i.e., in the range 100,000 to 999,999) and then constructs and outputs the integer whose digits are in the reverse order of the input. For example, if 247315 is input, then 513742 is output.
SOLN : Here are two versions of this, one with an array, and one without.

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

int main()
{
    int numb[6];
    int n;
    cout << "\nEnter a 6 digit number: ";
    cin >> n;
    for(int i = 0; i < 6; ++i)
    {
            numb[i] = n%10; // least signif. digit;
            n /= 10; //truncate 1's place
    }
    cout << "\nThe reverse of your number is ";
    for(int i = 0; i < 6; ++i) cout << numb[i];
    cout << endl;
    return 0;
}
```

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

int main()
{
      int n_rev = 0;
      int n;
      cout << "\nEnter a 6 digit number: ";
      cin >> n;
      for(int i = 0; i < 6; ++i)
      {
         n_rev += n%10; //least signif
         n_rev *= 10; //shift place val
         n /= 10; //truncate 1's place
      }
      cout << "\nThe reverse number is ";
      cout << n_rev << endl;
      return 0;
}
```

7. Write the output of the program below:

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

int main()
{
    double x = 3.0;
    int n = 5;
    double y = 1.0, z = x;
    int i = n;
    cout << "z = " << z << endl;
    while (i > 0) // INVARIANT: y*pow(z,i) == pow(x,n)
            if (i%2 == 0) {
                    z *= z;
                    i /= 2;
                    cout << "z = " << z << endl;
            }
            else {
                    y *= z;
                    --i;
                    cout << "y = " << y << endl;
            }
            cout << x << "^" << n << " = " << y << endl;
}
```

SOLN:

```
z = 3
y = 3
z = 9
z = 81
y = 243
3^5 = 243
```

8. What's going on in the code below? Explain. Note that the output of the program is shown below.

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

int main()
{
    unsigned long long c = 0;
    for (int i = 0; i < sizeof(unsigned long long) * 8; ++i) {
            unsigned long long d = 1;
            for(int j = 0; j < i; ++j) d *= 2;
            c += d;  //c |= 1 << i;
    }
    cout << c << endl << c+1 << endl;
    unsigned long long f = ULLONG_MAX;
    cout << "\nf = " << f << endl << f+1 << endl;
}
```

Here's the output:

```
18446744073709551615
0
f = 18446744073709551615
0
```

SOLN:

An unsigned long long is a 64 bit number whose maximum value is $2^{64} - 1 = 18446744073709551615$.

The nested for loops

```
for (int i = 0; i < sizeof(unsigned long long) * 8; ++i) {
        unsigned long long d = 1;
        for(int j = 0; j < i; ++j) d *= 2;
        c += d;  //c |= 1 << i;
}
```

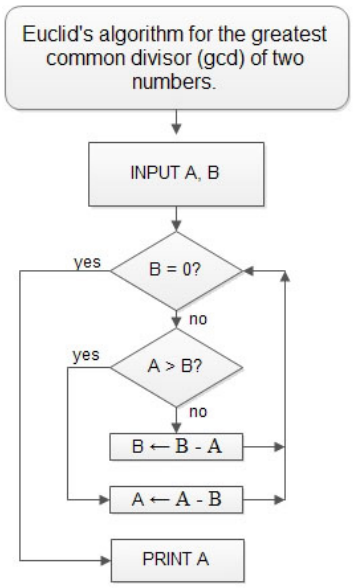
computes this as $\sum_{i=0}^{63} 2^i = \frac{2^{64}-1}{2-1}$ maximum by summing 64 (`sizeof(unsigned long long) = 8`) successive powers of 2. Then this is verified and the wrap-around nature is demonstrated by printing this value and one more than this value, which wraps around to 0.

A simpler way of doing this is demonstrated by the use of `ULLONG_MAX` defined in the `climits` library.

9. Write a C++ program to implement the algorithm described by the flow chart at right.

SOLN:

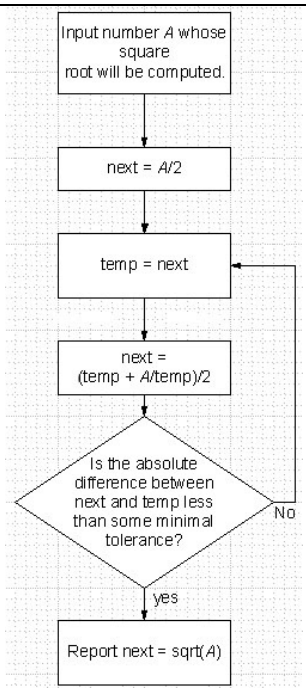
```
#include <iostream>
using namespace std;

int main()
{
        int A, B;
        cout << "\nEnter two positive and the gcd "
             << "will be computed by Euclid: ";
        cin >> A >> B;
        while(B != 0)
        {
                if(A > B) A -= B;
                else B -= A;
        }
        cout << "\ngcd = " << A << endl;
}
```



10. Write a C++ program to implement the algorithm described by the flow chart below.

SOLN:

```
#include <iostream>
using namespace std;
double abs(double x) { return x > 0 ? x : -x; }
int main()
{
        double A, temp, next, toler = 1e-10;
        cout << "\nEnter a positive number and this"
                <<  "program will use the "
                << "\nBabylonian algorithm to compute"
                << "the square root: ";
        cin >> A;
        temp = A/2.;
        next = (temp + A/temp)/2.;
        while(abs(temp-next)>toler)
        {
                temp = next;
                next = (temp + A/temp)/2.;
        }
        cout << "\nsqrt(" << A << ")=" << next<<endl;
}
```



11. Suppose the file mystery.txt contains the text below:

The whole thing starts about twelve, fourteen or inbetween.

What is the output of the program listed below:

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

int main()
{
        string word;
        int i = 0;
        ifstream read("mystery.txt");
        while(read >> word)
        {
                if(i < word.length())
                        cout << word[i];
                ++i;
        }
        cout << endl;
}
```

SOLN: The code prints the *n*th letter from the *n*th word, provided the word has that many letters.

```
The
whole
thing
starts
about
twelve,
fourteen
or
inbetween.
```

As you can see from the bold letters above, this results in "Thirteen" being printed.