CSC112 Spring 2011
Fundamentals of Computer Science
Practice Problems

1. What is the difference between a local variable and a global variable.

   Answer: A global variable is declared outside of any function, including the main. A global variable is accessible from any function. A local variable is either 1) a formal parameter of a function, or a variable declared within a function. If there is a local variable with the same name and a global variable, then the local variable will "shadow" the global variable. I.e., references within the function will refer to the local variable.

2. Write a short program that reads numbers from the standard input stream (cin) up to end-of-file and prints the largest and smallest numbers in the input set. **Answer:**

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

int main()
{
   int u, large, small ;
   bool loop ;

   cout << "Begin entering numbers, end with control-D" << endl ;
   cin >> u ;   // Get the first number.
   if ( cin.fail() ) {
       cerr << "Input does not appear to be a number." << endl ;
       exit(1) ;
   }
   if ( cin.eof() ) {
       cerr << "EOF before first number." << endl ;
       exit(2) ;
   }
   small = u ; large = u ; loop = true ;
   while (loop) {    // Get additional numbers.
      cin >> u ;
      loop = ! cin.eof() ;
      if ( loop ) {
         if ( cin.fail() ) {
            cerr << "Input does not appear to be a number." << endl ;
            exit(3) ;
         }
         if ( small > u ) small = u ;
         if ( large < u ) large = u ;
      }
   }
   cout << "Largest is " << large << endl ;
   cout << "Smallest is " << small << endl ;
}
```

3. Write a short program that reads hexadecimal digits up to end-of-line and prints the equivalent base 10 number.

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

int main()
{
   bool loop ;
   char ch   ;
   int u ;

   u = 0 ; loop = true ;
   while ( loop ) {
      cin.get(ch) ;
      loop = ch != '\n' ;
      if (loop) {
         if ( ('0' <= ch ) && ( ch <= '9' ) ) {
            u = u * 16 + (ch - '0')  ;
         }
         else if  ( ('a' <= ch ) && ( ch <= 'f' ) ) {
            u = u * 16 + (ch - 'a' + 10 )  ;
         }
         else {
            cerr << "Input character is not a hexadecimal digit." << endl ;
            exit(1) ;
         }
      }
   }
   cout << "Base 10 number is: "  << u << endl ;
}
```

4. What is the 8-bit base 2 representation of the (base 10) number 37 ?

   Answer: 0 0 1 0 0 1 0 1

5. Using two's complement representation, what is the 8-bit representation of the base 10 number -37 ?

   Answer: 1 1 0 1 1 0 1 1

6. What would the following program print ?

```
#include <iostream>
using namespace std ;
int main()
{
    cout << 1 + ~a  << endl ;
}
```

   Answer: -12

7. What would the following program print ?

```
#include <iostream>
using namespace std ;
int main()
{
    cout << (25 & -9)  << endl ;
}
```

Answer: `17`

8. The program in problem 7 works, but the one given below does not compile. I.e., g++ prints an error message. Why ?

```
#include <iostream>
using namespace std ;
int main()
{
    cout << 25 & -9 << endl ;
}
```

Hint: If I name the program file "u.cc", then the error message is:

```
u.cc: In function ‘int main()’ :
u.cc:5: error: invalid operands of types ‘int’ and ‘<unresolved overloaded
function type>’ to binary ‘operator<<’
```

Answer: The error here is that the operator precedence does not cause the bit-wise "and" to be performed first. The stream operator is higher precedence, so the compiler "tries" to translate the sub-expression `-9 << endl` . But, the left argument to the stream operator is an integer, and this operator requires a stream object (e.g., cout) as the left argument.

9. Write a function to accept an array of integers `a`, and the number of integers `n` stored in the array. Your function should return the largest integer in the array. The function header should be:

```
int find_largest( int a[], int n )
```

Answer: See code segment below.

```
int find_largest( int a[], int n )
{
   int i ;
   int largest ;

   largest = a[0] ;   //  Use the first element in the array as our
                      //  candidate for the largest element.

   //  The following loop goes through the rest of the array.
   //  If our candidate 'largest' is smaller than a[i], it can
   //  not be the largest, Right ?   In that case, we replace the
   //  value of 'largest' with a better candidate, i.e.,  a[i]
   //  Once the entire array is processed, we know that 'largest'
   //  contains the largest element in the array.

   for ( i = 1 ; i < n ; i++ ) {
      if ( largest < a[i] ) largest = a[i] ;
   }

   return(largest) ;   //  The return statement passes the value
}                      //  back to the caller.
```

10. Explain the difference between *pass by value* and *pass by reference* when passing parameters to a C++ function.

Answer: Pass by value copies the value of the actual parameter (in the caller) to the formal parameter (in the called function). Changes to the parameter in the called function does not change the value of the actual parameter in the caller.

Pass by reference causes changes to the formal parameter (in the function) to be reflected in the actual parameter (in the caller). There are two ways to do this.

We can use C++ pointers to create pass-by-reference behavior. In reality, when we pass the pointer, it is copied from the (pointer) actual parameter, to the (pointer) formal parameter. The function uses pointer reference to store into the memory location that is indicated by the pointer. This memory location is also the memory location of the corresponding variable in the caller.

We can also use C++ references to create pass-by-reference behavior. Using this language feature simplifies the syntax and partially automates the process of passing pointers described above. C++ references arrange the formal parameter and the actual parameter to refer to the same memory location.

11. Write a function to decide if a number is a perfect number. Note: A number is a **perfect** number if the number is the sum of all of it's divisors, excluding the number itself. For example, 6 is a perfect number because $1 + 2 + 3 = 6$ and 6 is divisible by each of $1, 2$, and 3. The function header is:

```
bool is_perfect( int k )
```

Answer:

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

const int MAX=100000 ;

bool is_perfect( int k )
{
   int sum_of_divisors ;
   int i ;

   sum_of_divisors = 0 ;
   for ( i = 1 ; i <= k/2 ; i++ ) {
       if ( (k % i) == 0 ) sum_of_divisors += i ;
   }
   return ( k == sum_of_divisors )  ;
}

int main()
{
   int k ;

   for ( k = 1 ; k < MAX ; k++ ) {
      if ( is_perfect(k) ) cout << k << " " ;
   }
   cout << endl ;
}
```

12. Write a function to accept an array of integers `a`, the number of elements in the array `n`, and a target value `x`. The function should return the position where `x` is found in the array `a`. If `x` is not found, then the function should return -1. The function header is:

```
int search( int a[], int n, int x )
```

Answer:

```
int search( int a[], int n, int x )
{
   bool found ;
   int i      ;

   found = false ;
   i = 0 ;
   while ( !found && (i < n) ) {
      found = a[i] == x ;
      if ( !found ) i++ ;
  }
  if (found) return i ;
  else return -1 ;
}
```