Review for midterm examination:

The examination will cover everything up to and including Linked List and recursion (tracing a recursive function). Here is a list of topics to study: the concept of type (including pointer, class, struct, enum, array), the concept of function (including parameter passing, local variable, static local variable, prototype, return value), the concept of overloaded function name or operator, the concept of class (including constructor, copy constructor, destructor, assignment operator, member operator overloading, getter and setter member functions or methods, public and private member, and scope resolution :: usage), the concept of pointer and array (including the array name is a pointer, the concept of deference a pointer, ways to deference a point, ways to provide value to a pointer, looping through elements in an array), the concept of memory management (including allocating space and deleting space, how to use the space as an array or an object of certain type), the concept of c-style string and its programming, the concept of linked list (including the -> notation, following the link, insert, delete, and traversal of a linked list), the concept of C++ string and vector types and their programming, and the ability to reason about C++ program (What does the code mean? What does the code do? Is there any potential problem (compile time or run time)?), and the ability to write program to solve problems similar to those that we have solved in Labs and Homework.

Sample review problems:

We should be familiar to all the problems we solved and the concepts we learned in our lectures, labs and homework. Problems are grouped into three sections: writing code, explaining and using concept, and understanding code.

1. Writing code.
   a. We are interested in having a function that takes in a c-style string as argument and return the length of its argument. Write the prototype of such a function.
   b. Let “int strlen ( const char * str );” be the prototype of the function in previous question. Provide the implementation of the function.
   c. We are interested in having a function that takes in two arguments of c-style string and concatenate the second string to the first. The function then returns the result, which is the first string. Write the prototype of such a function.
   d. Let “char * strcat( char * dest, const char * source );” be the prototype of the function in the previous question, implement the function.
   e. Implement “int strcmp( const char * s1, const char * s2 );” of which the definition is given as follows:

   Compare two strings

   Compares the C string s1 to the C string s2. This function starts comparing the first character of each string. If they are
equal to each other, it continues with the following pairs until the characters differ or until a terminating null-character is reached.

**Parameters**

- **s1**
  
  C string to be compared.

- **s2**
  
  C string to be compared.

**Return Value**

Returns an integral value indicating the relationship between the strings:

- A zero value indicates that both strings are equal.
- A value greater than zero indicates that the first character that does not match has a greater value in `s1` than in `s2`; And a value less than zero indicates the opposite.

f. Write code to define a data structure of a vector of char *. Read in 10 strings from the terminal and store them in the first 10 elements in the vector.

g. Write the code to implement “bool numeric (char * s)”, of which the definition is given as follows:

Decide if all the characters in C string s are digits 0 to 9.

Parameter: s is a C string.

Return value: true if all characters in s are digits and false otherwise.

h. Write the code to implement “bool check (char * s1, char c)”, of which the definition is given as follows:

Decide if character in c shows up in C string s1.

Parameter: s1 C string ; c is a character.

Return value: true if character in c shows up in s1 and false otherwise.

i. Write the code to implement “bool check (char * s1, const char * s2)”, of which the definition is given as follows:

Decide if all the characters in C string s1 are in C string s2.
Parameter: s1 C string of which characters are checked; s2 C string containing valid characters to be checked against.

Return value: true if all characters in s1 are in s2 and false otherwise.

j. Write a function that converts character digits 0 through 9 to integers 0 through 9. Here is the prototype “int convert(char c);”.

k. Write two overloaded functions that converts a string of digits to its equivalent integer. Here are the prototypes “int conversion(const char * s);” and “int conversion (string s);”.

l. Write the code to implement “char * replace (char * s1, const char * s2)”, of which the definition is given as follows:

Replace the substring from the last dot (‘.’) to the terminating null character in C string s1 by the value in C string s2.

This function modifies s1 if it contains a dot as suggested. If s1 does not contain dot, nothing is done to s1.

Parameter: s1 C string to be modified potentially; s2 C string for the replacement value.

Return value: s1

m. Write an efficient code to solve the Maximum Subsequence Sum Problem.

n. Write the code to implement “char * strreverse (char * s)”, of which the definition is given as follows:

Reverse the characters in C string s
This function puts the last character to the first position, the next to last character to the second position, and so on. So the values of s is reversed.

Parameter: s C string to be reversed

Return value: s

o. Search a linked list (singly linked).
p. Insert into a linked list.
q. Delete from a linked list.

r. Similar or identical problems appear in the labs and homework assignments.
2. Explaining concepts:
   a. How can we tell if a member function of a C++ class is a constructor?
      What is a constructor for?
   b. How can we tell if a member function of a C++ class is a copy constructor?
   c. Do we have to provide a copy constructor? Why or why not?
   d. How can we declare a constant member function? Why do we need constant member functions?
   e. Ordinary functions and member functions of a class may be overloaded.
      Use an example to explain the notion of function overloading.
   f. How can we tell if a member function of a C++ class is a destructor? What is a destructor for?
   g. The standard operators such as $<<$, $>$, $=$, and so on, may be redefined in a class or overloaded. How do we do that?
   h. We can specify whether a member field or a member function of a class is private or public. What are implications of these concepts?
   i. In our project, we have a user defined type Song. Consider the following prototypes of functions:
      i. void f(Song x);
      ii. void g(Const Song x);
      iii. void h(Song& x);
      iv. void p(const Song& x);
      Compare and contrast these functions.
   j. Compare and contrast C++ string and C style string.
   k. When should we use “this” and scope operator “::”?
   l. What is the type of array name variable?
   m. When we use the library class (or type) string and vector $<$ $>$, it seems that we can add as many elements (using push_back method) as we want. But we know an array usually has a fixed size and array is the underlying structure for string and vector $<$ $>$ . What is going on here and how do we implement such an impression of unlimited size of string or vector $<$ $>$ ?
   n. What is the difference between passing by value and passing by reference when we call a function? Why is this concept important? Illustrate.
   o. Why do we need a enum type?
   p. To overload $<$ $<$ and $>$ $>$ as output and input operators for a type defined by us, what do we need to do? If we do not do anything and use them, what would happen?
   q. Use an example to illustrate how an object or a variable gets associated with the three storage classes: stack, heap (free store), and global store. What are the implications?
3. Understanding code:
      ```c
      char x[3];
      char* p;
      x=p;
      ```
      ```c
      char x[3];
      char* p;
      p=x;
      ```
      ```c
      char x[3];
      char* p;
      p=x;
      p[0]='a';
      cout « p;
      ```
   d. Is there any problem? Explain.
      ```c
      char p[3];
      char q[] = "abc";
      strcpy(p,q);
      ```
   e. Is there any problem? Explain.
      ```c
      char p[4];
      char q[3] = {'a','b','c'};
      strcpy(p,q);
      ```
      ```c
      char *p;
      char q[] = "abc";
      strcpy(p,q);
      ```
   g. Is there any problem? Explain.
      ```c
      int n, a[10];
      for (n=1; n<=10; n++)
         a[n]=0;
      ```
   h. What is the output? Explain.
      ```c
      char p[10] = "abc";
      strcpy(p,p+1);
      cout « p;
      ```
i. What is the output? Explain.

    char p[10] = "abc";
    strcpy(p+3, p);
    cout << p;

j. Is there any problem? What is the output?

    struct Song {
        int x;
        char name[10];
    };

    void f(Song s) {strcpy(s.name, "abc")}

    // code in main program
    Song t;
    strcpy(t.name, "Hi");
    f(t);
    cout << t.name;

k. The Song type is given in the previous problem. What is the output?

    void f(Song& s) {strcpy(s.name, "abc")}

    // code in main program
    Song t;
    strcpy(t.name, "Hi");
    f(t);
    cout << t.name;

l. Is there any problem? What is the output?

    #define SIZE 4;
    int a[SIZE], k;
    int * p = a;
    for (k = 0; k < SIZE; k++)
        *(k + p) = k*k;

    for (k = 0; k < SIZE; k++)
        cout << "a[" << k << "] = " << a[k] << endl;