CS4700 Homework #6
Note, these exercises may be done in groups of one, two, or three.

1) In a switch statement, would a standard jump table be an appropriate implementation for the following distribution of case constants? If not, explain an alternative you would consider.
   a) Twenty cases close together in value (say a range of 50)
   b) A thousand cases close together in value (say a range of 1050)
   c) Five cases over a wide range of values (say a range of 1000).
   d) A thousand cases over a wide range of values (say a range of 1,000,000).

2) Order of evaluation. While precedence and association controls the parse tree that is formed, the order of evaluation within the tree might not be specified by the language design. If there are no side-effects, the order of expression evaluation doesn’t matter, but with side effects, the results are different depending on the expression evaluation order. What possible values for sum could result?

```plaintext
function Doit( var K: integer ) : integer // K is passed by reference
begin
  K := K * 4;
  return K;
end;

i := 5;
sum := i * Doit(i);
```

3) Study the ILM example of call stacks on csilm.usu.edu. Go to Programming/PROGRAMMING LANGUAGES/Call Stack. Using the model shown in the ILM, draw the stack of activation records for the following C program. Show the dynamic and static links and fill in the local names in each activation record.
   a) Show the call stack at line 4 every time it is reached.
   b) Describe how variables r and x are found on lines 4 and 5 (during the execution of p)

```plaintext
1. int x;
2. void p(void)
3. { double r = 2;
4.   cout << r;
5.   cout << x;
6. }
7. void r(void)
8. { x = 1;
9.   p();
10. }
11. void q(void)
```
12. { double x=3;
13.   r();
14.   p();
15. }
16. main()
17. { p();
18.   q();
19.   return 0;
20. }

4) We use the term *display* as follows: an array of all the accessible activation record addresses. So if A calls B calls C calls D, the stack (growing upwards) is shown below. The display points to all the activation records which could be consulted for non-local variables. In traditional static scoping, all variables are stored in an activation record on the stack. How are they represented in the display method? Write the pseudocode which shows how the display is updated when a procedure is called. Write the pseudocode which shows how the display is updated when a procedure is exited.
5) For dynamic scope, **deep access** requires that to find a non-local variable, dynamic links are used to search back in the stack for the most recent activation record that contains space for that variable. Note that this requires that it be possible to tell which variables are stored in each activation record; this is more natural for languages that are **interpreted** rather than being compiled. Note also that the number of dynamic links that must be followed cannot be determined at compile time; in fact, a different number of links may be followed at different times during execution. Trace the execution of the program given. Note that method P includes a use of non-local variable x. Begin execution at S(). Each time P is called, how many dynamic links must be followed to find the activation record with an x?

```c
void P() { write x; }
void Q() {
    x = x + 1;
    if (x < 23) Q();
    else P();
}
void R() {
    int x = 20;
    Q();
}
void S() {
    int x = 10;
    R();
P();
}
```