University of Toronto

Faculty of Arts and Science

December Examinations 2002

CSC 270H1F

Duration: 3 hours

Aids allowed: one $8\frac{1}{2} \times 11$" aid sheet (both sides).
Non-programmable calculators are permitted.

Last name:

First name(s):

Student number:

Lecture section: circle day or evening

There are 11 pages and each is numbered at the bottom. Make sure you have all of them.

Note: Answers not in the correct space will not be graded unless a note in the correct space says "see page ..." and the answer on that page is clearly labelled with the question number.

When you write code in C or C++, comments are not required, but good programming style is expected.

Do not write anything in the following table:

	value	grade		value	grade
1	10		6	10	
2	10		7	5	
3	10		8	10	
4	5		9	10	
5	10		10	10	
subtotal			subtotal		
			total	90	

1. [10 marks] Write a C or C++ function which returns the sum of an array of integers. It should work for any size array. Your task includes deciding upon the parameter list. Write the complete function.

2. [10 marks]
a. Use Newton's method to find a root of the function

$$f(x) = x^3 + 4x^2 - 5$$

Start with an x approximation of 3. Show your work for three iterations of Newton's method.

b. Assuming that the root you are converging on is 1, calculate the absolute error and the relative error, after your three iterations above.

3. [10 marks] The natural log of x (written $\ln x$) can be defined for all positive numbers as:

$$\ln x = \int_1^x \frac{1}{t} dt.$$

Write a C or C++ function that computes $\ln x$ using the above formula and one of the numerical integration techniques discussed in this course. (You may not use any math library functions which compute logarithms or exponentials.) Identify one possible source of numerical error, and indicate the step(s) you took to reduce it in your code.

4. [5 marks] Here is the code from the course notes that detects cycles in an undirected graph:

```
initialize num[v] to 0 for all v
initialize edgeSet to the empty set

cycleDetectionDFS(v)
    num[v] = i++;
    for all vertices u adjacent to v
        if num[u] == 0
            add edge (u,v) to edgeSet
            cycleDetectionDFS(u)
        else
            cycle detected
```

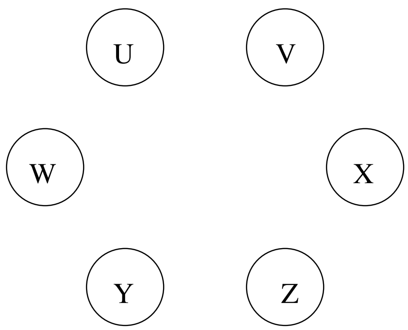
Note that the algorithm presented would consider *any* undirected edge to be a cycle (which is consistent with the somewhat strange definition of a cycle in the readings)—if there is an edge (u, v), then the path u, v, u is considered to be a cycle.

Rewrite the the above undirected graph cycle detection algorithm below so that it only detects cycles $v_j, \ldots, v_k$ where $v_k == v_j$, none of the other vertices repeat, and $j - k > 3$.

5. [10 marks] Given the weighted graph represented by the following adjacency lists, where the weights are in parentheses:

U	V(1)		
V	W(5)		
W	U(10)	Y(6)	Z(3)
X	V(4)	W(12)	
Y	U(3)		
Z	X(7)	Y(2)	

a. Draw the directed graph.



b. Using Dijkstra's algorithm, find the shortest paths from X to all other nodes in the graph. Give node X the number zero, then number the remaining vertices in the order that they are removed from the tobechecked set. (That is, the order that the nodes are settled.)

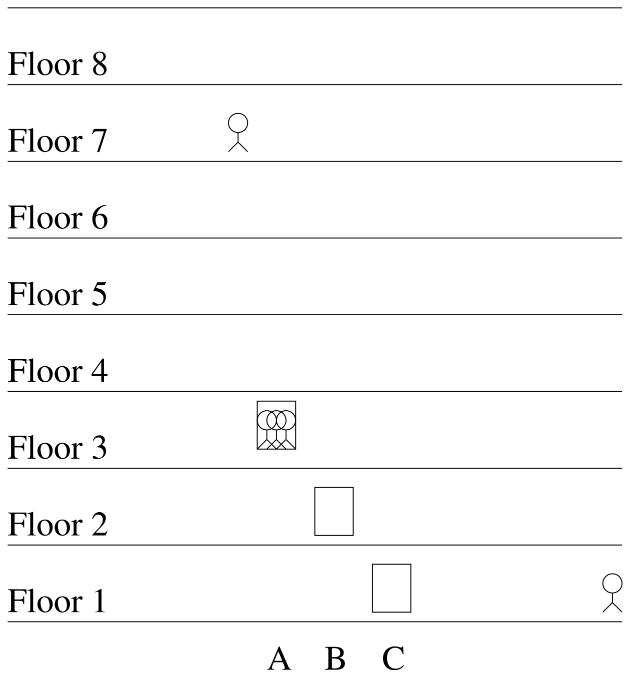
6. [10 marks] You have been hired to write a simulation program to model the weather. Among other requirements, you must choose an amount of rainfall per 24-hour period. You are told that days either have rainfall or do not, and the probability of rainfall in any given 24-hour period is 0.45. You are also told that for a 24-hour period which does have rainfall, the amount of rainfall follows a uniform distribution with a range of 1 centimetre to 3.5 centimetres inclusive.

Write a C or C++ function to return the amount of rainfall for a 24-hour period.

7. [5 marks] Give an example of a simulation topic for which a time-driven simulation would be more appropriate than an event-driven simulation, and state the difficulty with or disadvantage of using an event-driven simulation for that situation.

8. [10 marks] The control program for an elevator bank is not always simple.

Here is a bank of three elevators. Elevator 'A' is full of people, travelling upwards; elevators 'B' and 'C' are empty and stopped. A person on the seventh floor is just pressing the 'up' button to call for an elevator. A person on the first floor is walking towards the elevators and will press the call button in a minute.



What should happen when that person on the seventh floor presses the button?

There are many different strategies. A bad strategy would be to send all elevators to pick up the passenger on floor 7, thus making that person on floor 1 wait a long time until one of them comes back. A better strategy would be to have elevator A pick him up on the way by, since it's already in motion and nearer. Perhaps an even better strategy would be to send the closest *idle* elevator to pick up the passenger, so that they don't have to wait for all the stops which elevator A is going to make.

The point is that there are many possibilities and it is difficult to see what is best. The optimal strategy may be a combination of the above strategies.

To figure out what algorithms for elevator scheduling are better, we write a simulation.

This simulation contains an event queue containing, among other things, objects of type "elevator arrival" and objects of type "passenger arrival". They are members of classes which are derived from base class "event".

Sketch most of the contents of these three class declarations, in C++, on the next page.

8, your answer:

```
class event {
```

9. [10 marks] The five most frequent words in *Alice in Wonderland* are listed below. Word K_i has frequency f_i.

i	K_i	f_i
0	a	629
1	and	848
2	it	591
3	the	1629
4	to	726

You want to construct a binary search tree with keys comprised of these words, ordered alphabetically. In addition, it must be an optimal BST structure; that is, one where the sum of all the $d_i \times f_i$ is minimized, where d_i is the depth of the ith node, and f_i is the frequency of the ith node's key. The depth of the root is 1, and we add 1 for each subsequent edge you need to traverse.

a. Fill in the table $C[i][k]$ below, where $C[i][k]$ is the minimal cost (sum of the $d_i \times f_i$) of a BST comprised of nodes containing keys $K_{i+1}, \ldots, K_{k-1}$. The cost added by selecting node j as the root of a (sub)tree is $w(i, j, k) = f_{i+1} + \cdots + f_{k-1}$, and the recurrence is:

$$\text{For } k \geq i+2,\ C[i][k] = \underset{\substack{\text{for } j \text{ such that} \\ i < j < k}}{\text{MIN}} \left(C[i][j] + C[j][k] + w(i, j, k) \right)$$

$$\text{For } k < i+2,\ C[i][k] = 0.$$

We have filled in the first couple of entries on the diagonal $C[i][i+2]$ to get you started:

	1	2	3	4	5
–1	629				
0	0	848			
1	0	0			
2	0	0	0		
3	0	0	0	0	

b. Draw a BST that satisfies the minimal cost for the keys in the previous part.

10. [10 marks] The problem of making change using the minimum number of coins can be implemented using dynamic programming. For the fictitious currency "Δ", write an algorithm in pseudo-code (and/or C or C++) to use dynamic programming to solve the problem of finding the minimal number of coins which sums to each of the values from $\Delta 0$ to $\Delta 100$, using coins of denomination $\Delta 1$, $\Delta 7$, $\Delta 25$, and $\Delta 30$.

(Note that, for example, $\Delta 32$ is most minimally represented with a 25 and a 7, rather than using the $\Delta 30$ coin—you can't just take the largest coin which fits, as you can with Canadian money. The full dynamic programming minimization technique is required in the case of the "Δ" currency.)