Coursework 2: Binomial Trees

- Marks will be awarded for the structure of your code, including comments and sensible choices for variable names.
- You should submit answers to questions 1, 5, 6 and 7 and a copy of your codes by 4:00pm on Monday 24th November 2008.
- You will also be expected to demonstrate working versions of your codes in the examples class on Monday 24th November 2008.

The aim of this project is to construct binomial trees using objects in C++. A binomial tree can be used to model the behaviour of an asset over time. The idea is that after a fixed time interval the asset value, V, can increase or decrease by given amounts each with a given probability. For example, in the case of a symmetric tree, the asset value increases by 1 with probability 0.5 and decreases by 1 also with probability 0.5. The first three (time) levels of such a tree are shown below:

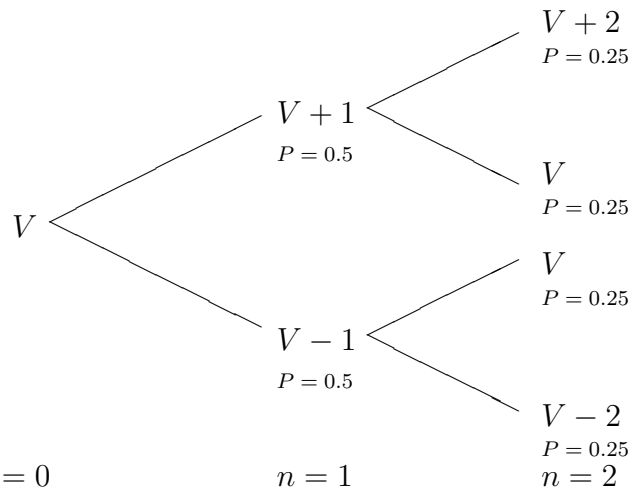

Figure 1: The first three levels of a symmetric binomial tree. The expected value of the asset at each level is always V; and yet the probability that the asset is equal to the expected value is 0 at level 1 and 0.5 at level 2.

The expected value of the asset at each level is the sum of the value multiplied by the probability over all "nodes" at that level.

$$\text{Expected value } (\bar{V}) \text{ at level } 1 = 0.5(V+1) + 0.5(V-1) = V,$$

$$\text{Expected value } (\bar{V}) \text{ at level } 2 = 0.25 * (V + 2 + V + V + V - 2) = V.$$

The probability that the asset takes a particular value $\hat{V}$ at a given level is the sum of the probabilities at nodes that take the value $\hat{V}$.

$$\text{P(asset has value V at level 2)} = 0.25 + 0.25 = 0.5.$$

In this project, we shall build a `Tree` object that consists of a collection of `Node` objects, each of which stores the value of the asset, the probability of reaching that value and the level of the `Node`. Note that the `Tree` **has** `Node`s, but **is not** a `Node`, so the `Tree` class should **not** inherit from `Node`. The `Tree` will also store "global" properties such as the initial value of the asset, the amounts by which the asset increases or decreases and the probabilities of increase and decrease.

Questions

1. Extend the symmetric binary tree shown above to four levels and calculate (by hand) the probability that the asset takes the expected value, V, at level four. [2 marks]

2. Write a `Node` class in C++ that represents a single point in a binary tree. Your class should contain data representing the value of the asset, V; the probability that the particular node is reached, P; and the level in the tree, n. Note that the "root" of the tree is conventionally labelled as level 0. Make sure that the constructor initialises all your member data. Your class should also contain a member function `output()` that prints the value, the probability and the level stored in the `Node` to the screen. [5 marks]

3. Write a `BinaryTree` class in C++ that can store any number of `Node`s. One (recommended) way to do this is to use dynamic memory allocation, *i. e.* by storing a pointer to a `Node`.[1] The constructor should take arguments representing the initial value of the asset, the probabilities of increase and decrease, and the amounts by which the asset increases and decreases. The `BinaryTree` constructor should then construct a single (root) `Node`.

   Make sure that you also write a destructor to clean up any memory that you might have allocated in the constructor.

   Finally, create a single symmetric `BinaryTree` in the `main` function of your program. The initial value of the asset should be 100, with equal probabilities (0.5) of increase and decrease of 1 unit.

   [5 marks]

4. Add a member function to the `BinaryTree` class that creates the next level in the tree. You should plan to store only the final level of the tree, so you can replace the existing `Node`s by the newly-created `Node`s.

   Test your function by writing a loop in `main` that calls this new member function four times and comparing the results of the `Node::output()` function called for all the `Node`s on the final level of the tree with your hand-calculations from question 1.

   [5 marks]

[1] Alternatively, you could use a fixed-size array of `Node`s.

5. Add two new member functions to the `BinaryTree` class. One function should calculate the expected value of the asset at the final level of the tree: $\bar{V}$. The other function should calculate the probability that the asset takes a particular value $\hat{V}$ at the final level of the tree.

   Test your functions by calculating the expected value, $\bar{V}$, and the probability that the asset takes the expected value, $P(\text{asset value is } \bar{V})$, for the final level of a four-level symmetric binary tree. What are these values?

   [4 marks]

6. Assuming an initial asset value of 100, use your program to find the expected value of the asset at levels 0 to 10 of the tree for $p_{up} = 0, 0.25, 0.5, 0.75$ and 1. Generate a table of this data, see Table 1. N.B. A print out of a file containing this data is acceptable. [6 marks]

Level / p_{up}	0	1	2	3	4	5	6	7	8	9	10
0.0	100										
0.25	100										
0.5	100	100	100								
0.75	100										
1	100										

Table 1: Table of expected value of the asset at different levels of a symmetric binary tree for given probabilities of an increase of 1% of the initial value.

7. Determine whether it is possible to generate the same table as in question 6 (same expected values at each level of the tree) by keeping $p_{up} = p_{down} = 0.5$ but by varying the values by which the asset can increase and decrease between levels. Explain your reasoning. If is is possible, use your code to produce the equivalent table indicating the values of the changes in the asset value that correspond to each row. [6 marks]