Exam 2

MIDTERM EXAM 2

Note: Midterm 2 contains 5 questions. Questions written on paper should be photographed and uploaded as JPEG or PDF. Question 5 should be submitted as C++ source file in a separate folder.

Question 1:

Alice sends messages to Bob using only eight voiced consonants from this alphabet: {B, D, G, J, N, R, V, Z}. Each consontant is encoded as a sequence of bits (0s and 1s) using the binary tree shown below:

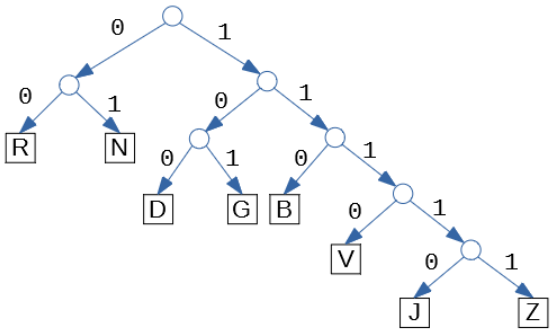

Fig. 1: A Binary Tree used by Alice to encode 8 consonant letters.

For example, VZJ is encoded as 1110.11111.11110 (V becomes 1110, Z becomes 11111 and J becomes 11110). Each code is obtained by following the edges from the root to the respective leaf in this tree.

(A) Show how Alice can encode the following 8-letter word: BRBDNGNG. How many bits does it use? (Please separate the codes of individual letters with dots for better readability.)

(B) In Alice's language the probabilities of letters are the following:

R	N	D	G	B	V	J	Z
32%	28%	14%	11%	9%	4%	1%	1%

Find the expected value of the total number of bits (0s and 1s) that she uses in order to send a random 10-letter word to Bob. (The letters in a random word are chosen independently according to the probabilities given above).

Question 2:

Some binary tree T has exactly 32 internal nodes.

(A) Can tree T be a full binary tree? (In a full binary tree every node has either two children or no children at all.) Can tree T be a perfect binary tree? (In a perfect binary tree all leaves have the same depth.)

(B) What is the largest and the smallest value for n – the total number of nodes in the tree T? Explain your estimates.

(C) What is the largest and the smallest value for h – the height of T? Explain your estimates.

Question 3:

Minimum Priority Queue has this ADT (Abstract Data Type):

PQ.getEmpty()	$\Theta(1)$	// initialize `PQ` to an empty priority queue
void PQ.insert(E item)	$\Theta(\log_2 n)$	// insert `item` into the priority queue `PQ`.
E PQ.min()	$\Theta(1)$	// return an item with minimum key value, do not modify `PQ`.
void PQ.removeMin()	$\Theta(\log_2 n)$	// remove an item with minimum key from `PQ`
int PQ.size()	$\Theta(1)$	// return the number of items in the priority queue `PQ`

Every function in this ADT has its time complexity written in the 2nd column – it corresponds to the heap implementation.

Consider the following pseudocode. Denote the number of items in the original list `L` by n (assume that $n \geq 10$ and all items in this list have different keys).

PROCESSLIST(L):
    PQ.getEmpty()
    foreach *item* **in** *L*:
        PQ.insert(item)
    while *PQ.size()* > *5*:
        PQ.removeMin()
    return *PQ.min()*

(A) Describe in English what does the function PROCESSLIST(L) return.

(B) Express the time complexity of this algorithm in Big-Θ notation: Find a function $g(n)$ such that the time complexity of PROCESSLIST(L) is $\Theta(g(n))$.

Question 4:

We have a 1-based array with 11 elements: $A[1], \ldots, A[11]$. We want to sort it efficiently. Consider the following Merge sort pseudocode:

MERGESORT(A, p, r):
1 **if** $p < r$
2 $q = \lfloor (p + r)/2 \rfloor$
3 MERGESORT(A, p, q)
4 MERGESORT($A, q + 1, r$)
5 MERGE(A, p, q, r)

Assume that initially you call this function as MERGESORT(A,1,11), where $p = 1$ and $r = 11$ are the left and the right endpoint of the array being sorted (it includes both ends).

What is the total number of calls to MERGESORT for this array (this includes the initial call as well as the recursive calls on lines 3 and 4 of this pseudocode).

Question 5:

Complete the C++ program that converts a tree into a string using function `toString()`. A tree can be built from two types of objects: objects of class `Leaf` and objects of class `Internal`. They are both inherited from a common parent class `Node`. All nodes have attribute `label`. Moreover, `Internal` nodes have attribute `children` of type `list<Node*>` – it is a list of pointers to the child nodes (either leaves or other internal nodes).

```
class Node {
public:
    string label;
    virtual string toString() = 0;
};

class Leaf : public Node {
public:
    Leaf(string arg) { label = arg; }
    virtual string toString()
    {
      // TODO: Insert your code here
    }
};

class Internal : public Node {
public:
    list<Node *> children;
    Internal(string arg, list<Node *> ch)
    {
        label = arg;
        children = ch;
    }
    virtual string toString()
    {
      // TODO: Insert your code here
    }
};

int main() {
    Node *tree1Root = new Leaf("AAA");
    cout << tree1Root->toString() << endl << endl;

    Node *tree2Root = new Internal("AA",
        {new Leaf("BB"), new Leaf("CC")});
    cout << tree2Root->toString() << endl << endl;

    Node *tree3Root = new Internal("A",
        {new Internal("B", {
            new Internal("C", {new Leaf("D"), new Leaf("E")}),
            new Internal("F", {new Leaf("G"),
                new Internal("H", {new Leaf("I"), new Leaf("J")})})}),
        }),
        new Internal("K", {new Internal("L", {new Leaf("M")})})});
    cout << tree3Root->toString() << endl
         << endl;
}
```

A tree that is a leaf is converted to a string: the value of `label`. A tree that is an internal node is converted to a string as the parent's label followed by all the subtrees under that parent – they are all separated by single spaces and enclosed in parentheses.

Here is the expected output from the program:

```
AAA

(AA BB CC)

(A (B (C D E) (F G (H I J))) (K (L M)))
```

In this task you can complete the functions `toString()` and possibly make other changes. You should not modify the method `main()`; your code should preserve the inheritance relations between `Node`, `Leaf` and `Internal`. Solutions that use "hard-coded" output that does not depend on the values and structures defined in `main()` will not be considered valid.