

CPSC 3740—Fall 2023

Assignment 2

Due: October 17, 2023 11:55pm

Instructions

All written responses must be submitted in Crowdmark. The programming exercises must be submitted on Moodle.

Written Problems

1. (2 marks) A Boolean value can be represented using one bit, but many programming languages store them in individual bytes. Name one advantage and one disadvantage of using one bit to store a Boolean value.
2. (8 marks) In C++, there are additional keywords that modify variable declarations. These include **extern**, **static**, **volatile** and **mutable**. Find online resources on these keywords and read them.
 - What does it mean when a global variable is declared as **extern**? Why is that needed?
 - What does it mean when a global variable is declared as **static**? Why is that needed?
 - What does it mean when a variable is declared as **volatile**? Why is that needed?
 - What does it mean when a member variable of a class object is declared **mutable**? Why is that needed?
3. (4 marks) C/C++ allows the use of pointers, but Java only supports reference variables to refer to heap-dynamic variables. What are some advantages of Java's approach vs. C/C++? What are some disadvantages?

4. (10 marks) Consider the following program in C-like syntax:

```
void f1();
void f2();


int main()
{
    int x, y;
    ...
}

int f1()
    int a, b, x;
    ...
}

int f2()
{
    int b, c, x, y;
    ...
}
```

If the programming language uses dynamic scoping, what are the variables that are visible in the last function in the following call sequences? Give not only the names of the variables, but also where they are declared.


- (a) $\text{main} \rightarrow \text{f1} \rightarrow \text{f2}$
- (b) $\text{main} \rightarrow \text{f2} \rightarrow \text{f1}$

-  5. (4 marks) Assume that the size of `short` is 2 bytes in C++. A multidimensional array is declared as

```
short A[17][22][12];
```

Compute the byte offset from the starting address of `A` for the following elements:

- (a) `A[3][7][5]`
- (b) `A[10][10][10]`

-  6. (5 marks) Assume the following 2-dimensional array has been declared in C++:
- ```
short A[1234][5678];
```

- (a) If the elements are examined in row-major order (one row at a time), how does the address of the element being examined change from one element to the next?
- (b) If the elements are examined in column-major order (one column at a time), how does the address of the element being examined change from one element to the next?
- (c) Why is it more efficient to examine the elements in row-major order than column-major?

## Programming Exercises

In this assignment, you will implement some functions in Racket to deal with trees. Write your solution in the file `as2.rkt` and submit it on Moodle.

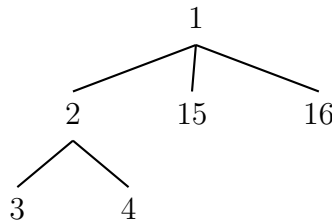
Nested lists can be used to represent trees as follows:

- The empty tree is represented by an empty list.
- A non-empty tree is represented by a list in which the first element is the label of the root node, and the remaining elements are representation of the subtrees in the specified order.

For example,

```
(1 (2 (3 () ())) (4 () ())) (15 () ()) (16 () ()))
```

represents the following tree:



Notice that this representation allows any nonzero number of subtrees in each node.

We will assume that a leaf node will still have at least one subtree, but all subtrees will be empty trees. In addition, all node labels are integers.

1. (10 marks) The size of a tree is defined by the number of nodes in a tree. For example, the example tree above has a size of 6.

Write a function `tree-size` that will take a tree representation as described and return the size of the tree.

2. (10 marks) The level of a node in the tree is the number of nodes in the path from the root to the node (including itself). For example, the node 15 in the example tree above has a level of 2.

Write a function **prune-tree** that will take a tree representation as well as an integer **level**. It will return a new tree containing only nodes that have level not exceeding **level**.

The order of the nodes in the result list should be the same as the original order in the tree. You may assume that **level** is a non-negative integer.

3. (10 marks) Write a function called **count-tree** that will take a tree representation and an integer **target**, and determine how many times the element **target** is located in the tree.