# Assignment 1: SmartyPants C++ Trivia Game

Daniel Bueso-Mendoza 117005010

September 9, 2013

## 1 Problem Description

The purpose of this assignment was to create a simple C++ trivia game that takes a file as its input stream to create "Question" objects that are stored in STL containers of either vector or list. A more versed usuage of vector and list containers was another goal of the assignment. The best and worst case scenarios are provided for these data structures.

### 1.1 Program Design

An input text file consisting of 15 C++ multiple choice questions are read. A question, its answer choices, and the correct answer are stored into a single Question object. A Question object is denoted in a separate header file where its functions and parameters are formulated. The created Question objects are stored in either a list or vector. The program then selects 10 of the 15 Question objects to display to the user for the trivia game. The user responds with an answer to the question, if correct or incorrect the feedback is displayed instantly. At the end the user's score is displayed.

### 1.2 Data Structures Description

Vector and list are both part of the C++ standard template library (STL). Vectors are akin to a dynamic array with other operations such as a copy constructor, copy assignment, vector size, and others. A subscript operator is also allowed for use with vectors. Data is stored in a single block of memory (contiguous) linearly like an array. A list has some similarities to a vector. Lists can push/pop and insert elements into its container. Unlike a vector, a list can insert and push/pop elements from the front and back. A subscript operator is not allowed for use with lists. One can access the first and last element in a list using keyword *front* or *back* but for elements in between a pointer is utilized. Lists also store its elements in a linear manner but instead of it all being in a single chunk of memory, the storage can be spread around with one element location pointing to the next.

The best case scenario for both a list and vector is $O(1)$. This meaning that if one is lucky they may hit the element they want on their first try. Typically this is for the first

element in either data structure. The worst case for both vector and list is $O(n)$. Both containers store their elements in a linear form. If the element one is looking for is the last element in either structure, it will take $n$ number of iterations to reach it. Here $n$ refers to the container size.

### 1.3 Algorithm Description

The design of the SmartyPants trivia game began with devising a Question class. The Question class has three parameters: string question_statement, vector<string> choices, and an integer for answer. Question_statement stores the question to ask, choices holds the answer choices for a question, and answer carries the integer specifying the correct answer. A basic constructor is in this class as well as a call operator. A boolean operator was overloaded to compare two Question objects and compare if they are the same. The »operator was overloaded to take a file stream (ifstream) as its input. Inside of the »operator, parsing of the file takes place as well as Question object variable assignment. The «operator was modified to clearly display a question object and its elements.

The first step for the game portion of the program reads in an input text file of 15 C++ questions. These questions are stored in Question objects. Next a user is allowed to select either a vector or list container. The Questions objects are then pushed onto either a list (listQuestions) or vector (vectQuestions). A separate vector called *guesses* contains the indexes for a question object stored either in listQuestions or vectQuestions. Guesses has its values randomly scrambled. This is later utilized to select Question objects randomly and display to the user. Once a question is displayed, it then waits for user input. If the user input matches the current Question object's answer parameter then *That's right you're smart!* is shown and a pass variable is incremented. This continues for 10 questions total and a score (out of 10) is presented at the end.

### 1.4 C++11 Feature

A single C++11 feature was incorporated in this assignment. Here is what it is in the code:

$$list < Question >:: iterator\, it2 = next(listQuestions.begin(), *it); //it2\, points\, to\, the\, element\, at\, index\, *it$$

The keyword *next* is part of the new STL C++11 (std). Here next takes std::list and creates a pointer to another element in list depending on the value of *it. The index *it is taken from the vector guesses.

### 1.5 Debugging

Debugging the code was done using *cout* statement of variables as well as using the built-in debugger in Xcode. Xcode allows one to set breakpoints (markers to pause the program)

and let the user step through the code. Xcode also enables the user to mouse over variables in the code while debugging to see what it contains.