# COMP 2404/2004 -- Assignment #4

## The Tortoise Chronicles: The Prize

Due: Thursday, April 5, at 23:59 PM

Collaboration: You may work in groups of no more than two (2) students

## Background

Our heroes' master plan is finally coming together. The Dread Pirate Harold (formerly known as handsome Prince Harold the Hare) and his Dread First Mate Timmy Tortoise have succeeded into luring King Humpernickel onto their pirate ship. Under the guise of an ordinary Tortoise and Hare, our dynamic duo has managed to fool Humpernickel into believing that they have locked up a pirate gang and are handing over their ship, the *Lost Soul*, to the sadly misinformed regent. In truth, Timmy and Harold, along with their crew, are planning to abduct King Humpernickel and deliver him to the Ninja Queen, in exchange for another, very important prisoner. Needless to say, King Humpernickel is in for quite a surprise.

Timmy Tortoise, an ace C++ programmer, is putting together a simulation of the abduction, to make sure that the plan is viable and victory is guaranteed. Your assignment is to help Timmy in coding the simulation of the battle between the King's soldiers and the pirate crew of dorcs, borcs and porcs, and the King's abduction at the hands of Timmy and Harold.

## Learning Objectives

In completing this assignment, you will:

- design a C++ program using:
  - basic software engineering principles, including encapsulation, extensibility and reusability
  - an established architecture design pattern, the Model-View-Controller (MVC)
- write a C++ program on Linux, using:
  - a linked list as a class template
  - an STL container class
  - virtual and pure virtual functions
  - polymorphism

## Submission

You will submit on WebCT, before the due date and time, the following:

- a UML class diagram (as a PDF file) that corresponds to your program design; your diagram will depict:
  - all class associations (composition and inheritance)
  - class names
  - class interfaces (i.e. class public data members and public member functions prototypes)
- a C++ program that meets all the functional and non-functional requirements described in the next pages; all the program source code must be archived together into one `tar` file, including the corresponding Makefile and a readme file describing how the user can compile, execute and operate your program

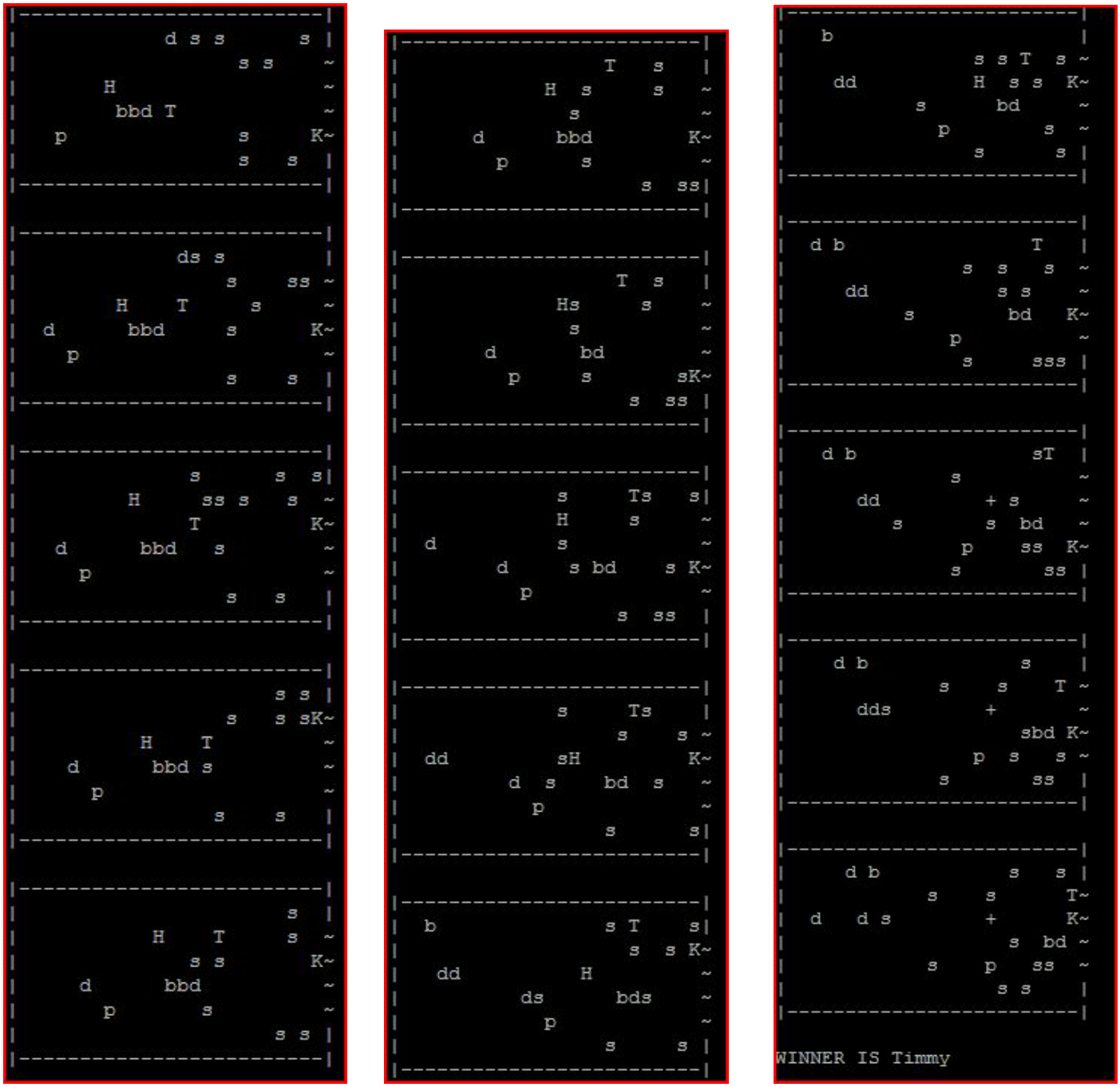
# Program Requirements:

## ➢ Functional requirements:

- The battle between pirates and soldiers takes place in the hold of the pirate ship. At start-up, your program displays the following:
  - The hold of the ship, represented as 25 characters wide (horizontal axis) and 6 characters high (vertical axis); the right-most edge of the hold represents a door in the middle four characters (see sample output)
  - Initially, the following warriors are shown:
    - Timmy (represented by his avatar 'T') and Harold (his avatar is 'H') are situated at the left-most edge of the hold, at different randomly generated vertical positions (use the random functions here).
    - The King (represented by his avatar 'K') situated at the right-most edge of the hold, at the midpoint on the vertical axis.
  - **Note**: The display **must** be stored as a 2-dimensional STL `vector`.
- Timmy and Harold each begin with 30 health points; Timmy has a strength of 5 points and an armour of 8 points, and Harold has a strength of 8 points and an armour of 5 points.
- **Note**: You will need a Warrior class, used as a base class for all types of warriors. Your program must store a list of all the warriors in the ship's hold, i.e. the *warrior list*. This list must be implemented as a linked list of Warrior pointers, and you must implement this list as a class template (do **not** use an STL container for this).
- The program iterates until Timmy or Harold abducts the King, i.e. until one of our heroes collides with the King (moves into the position occupied by the King), or until both heroes are dead.
- At each iteration:
  - With a probability of 1 out of 2, your program generates a new pirate:
    - A new pirate has equal probability of being a dorc, a borc or a porc. Dorcs are represented by the avatar 'd', borcs by 'b', and porcs by 'p'.
    - Pirates are initially situated on the left-most edge of the hold, at a randomly generated vertical position.
    - Pirates each begin with 20 health points, a strength randomly generated between 4 and 7 points, and an armour randomly generated between 2 and 4 points.
  - With a probability of 1 out of 3, your program generates a new soldier:
    - New soldiers are initially situated on the right-most edge of the hold, with a vertical position between the boundaries of the door to the hold.
    - Soldiers are bred to be nearly invincible, so they never lose health points and don't need them. Each has a strength of 15 points, and an armour of 3 points (King Humpernickel is kind of cheap that way).
  - Your program loops over each warrior in the warrior list. For each warrior, your program must invoke a function that **polymorphically** computes the warrior's next position. The warriors move according to these rules:
    - Timmy and Harold move one position to the right, with a vertical position randomly generated between the same vertical position, one position up, or one position down.
    - Pirates move in a straight line, exactly one position to the right, at the same vertical position.
    - Soldiers move one position to the left, with a vertical position randomly generated between the same vertical position, one position up, or one position down.
    - The King paces up and down in front of the door, moving up one position until he reaches the top position of the door, then reversing direction, until he reaches the bottom position of the door, then reversing direction, and so on.
    - If the new position computed for a warrior is out of bounds of the hold, the warrior doesn't move at all.
    - If the new position is occupied by another warrior, a collision ensues, and is handled as described below.
    - If the new position is not occupied, the warrior moves to that position.
    - Any pirates that have reached the right-most edge, or soldiers that have reached the left-most edge, disappear from the hold. They are removed from the warrior list.
  - The display is refreshed to show all the warriors from the warrior list, in their updated positions.

- Handling collisions:
  - If Timmy and Harold collide with each other, or with a pirate, nothing happens.
  - If two pirates collide with each other, nothing happens.
  - If a pirate or soldier collides with the King, nothing happens.
  - Soldiers are not used to fighting in dark, smoky spaces. If two soldiers collide with each other, they both die.
  - If Timmy, Harold, or a pirate collides with a soldier, that warrior's health is decreased by the damage inflicted by the soldier. The damage is equal to the soldier's strength minus the warrior's armour. When a warrior's health reaches zero, he dies.
  - Dead warriors are removed from the warrior list.
  - If Timmy or Harold dies, his position is permanently marked with a '+' to indicate his grave.
  - If Timmy or Harold collides with the King, that hero wins, the King is abducted, and the battle is over.

## ➢ Sample output (read top-down):



### ➢ Non-functional requirements:

- The program must display the output using the Curses character-based graphics library
- Design: the program must follow proper object-oriented design principles, including but not limited to:
  - encapsulation of data and functionality within classes, i.e. the principle of least privilege
  - easy modifiability and extensibility
  - Hint: you do not need any global variables; you only need global functions for main, the random number generator, and stream insertion operators -- all other functionality must be encapsulated within classes
- Implementation:
  - the program have all its classes organized in a Model-View-Controller (MVC) architecture
  - the program **must** use the `vector` class from the Standard Template Library (STL) for the output; the program must **not** use any classes from the STL for any other purpose
  - the program must store all the warriors in the hold in a linked list, implemented as a class template, with nodes separate from the data, which must be Warrior pointers; do **not** use an STL container for this
  - the program must explicitly deallocate all dynamically allocated memory – use `valgrind` to check for memory leaks
- Organization: the program must be correctly separated into source and header files
- Platform: the program must be written in C++, and it must compile and execute on the SCS lambda servers

# Grading

### ➢ Assignment grade:

- Your original grade will be computed based on how much of the functional requirements your program implements
- Your final assignment grade will be the total of your original grade and bonus marks, minus deductions

### ➢ Deductions:

- Any submission instruction or non-functional requirement that you do not follow will result in severe deductions, including but not limited to:
  - **100 marks** if:
    - the assignment is marked **Missed** in WebCT
    - any submitted files (including the Makefile) are missing, or corrupt, or in the wrong format
    - the code does not compile using g++ on the SCS lambda servers
    - the program does not use the Curses graphics library
  - **50 marks** if:
    - the design does not generally follow the principles of encapsulation and least privilege
    - the design does not follow a MVC architecture
    - global variables or global functions (other than main, RNG, and overloaded stream insertion operators) are used
    - STL containers are used, other than as prescribed
  - **20 marks** if:
    - the submission consists of anything other than exactly one diagram file and exactly one `tar` file
    - the program consistently crashes
    - the code is not correctly separated into header and source files
    - the readme file is missing or incomplete
  - **10 marks** for missing comments or other bad style (non-standard indentation, improper identifier names, etc)

### ➢ Bonus marks:

- Up to **10 extra marks** are available for fun and creative additional features