Background
University of Technology, Sydney

Faculty of Engineering & Information Technology
School of Software

31251 – DATA STRUCTURES AND ALGORITHMS
32510 - PRINCIPLES OF OBJECT ORIENTATED PROGRAMMING IN C++

AUTUMN 2011

ASSIGNMENT 1 DUE DATE
Week 13 - 1/6/2011 – 11.59pm

This assignment is worth 11% of the total marks for this subject.

Background

Connections is a strategy game for two players, designated *Player 1* and *Player 2*.

The Board

The game is played on a board of 4 x 6 squares that looks like this.

```
 ----------------------
|   |   |   |   |   |   |
|---+---+---+---+---+---|
|   |   |   |   |   |   |
|---+---+---+---+---+---|
|   |   |   |   |   |   |
|---+---+---|---|---+---|
|   |   |   |   |   |   |
 -----------------------
```

The Pieces

There are two types of pieces in the game - *crosses* and *plusses*. They look like this for each player.

Player 1 - cross `(x)` plus `(+)`
Player 2 - cross <x> plus <+>

For the purposes of the game each player is assumed to have an unlimited number of each piece.

Initial Setup

At the start of the game, the board will be set up thusly.

```
 -----------------------
|(+)|   |   |   |   |<+>|
|---+---+---+---+---+---|
|   |   |   |   |   |   |
|---+---+---|---+---+---|
|   |   |   |   |   |   |
|---+---+---|---+---+---|
|<x>|   |   |   |   |(x)|
 -----------------------
```

The Move

Starting with Player 1, each player alternates placing one of their pieces in any empty square on the board. Once a piece has been placed in a square it is no longer empty.

Connecting Pieces

If a *plus* piece is placed in one of the squares on the board then any of the squares immediately adjacent to the square are said to be connected to the piece. The following diagram illustrates the concept.

```
+---+---+---+
|   | c |   |
+---+---+---+
| c | + | c |
+---+---+---+
|   | c |   |
-------------
```

c = connected square

If a *cross* piece is placed in one of the squares on the board then any of the squares immediately diagonal to the square are said to be connected to the piece. The following diagram illustrates the concept.

```
+---+---+---+
| c |   | c |
+---+---+---+
|   | x |   |
+---+---+---+
| c |   | c |
-------------
```

Capturing an Opponents Piece

If a player has 3 or more pieces that are connected to the same square containing an opponents piece, then the opponents piece is changed to the same type of piece but belonging to the player. The following diagram illustrates.

```
+---+---+---+
|   |(+)|(x)|
+---+---+---+
|   |<+>|   |
+---+---+---+
|(x)|   |   |
-------------
```

In the example, Player 1 has 3 pieces that all connect to the square containing the plus piece belonging to player 2. Player 1 thus captures the piece to result in.

```
+---+---+---+
|   |(+)|(x)|
+---+---+---+
|   |(+)|   |
+---+---+---+
|(x)|   |   |
-------------
```

It is possible that capturing a piece may make it possible to capture another piece. If so, then that piece is captured. This is continued until all possibilities for capture are exhausted. The following diagram illustrates.

We start with this, with Player 2 to move

```
+---+---+---+
|<x>|<+>|   |
+---+---+---+
|(x)|(+)|   |
+---+---+---+
|<+>|<x>|   |
 ------------
```

Player 2 makes the following move

```
+---+---+---+
|<x>|<+>|   |
+---+---+---+
|(x)|(+)|<+>|
+---+---+---+
|<+>|<x>|   |
 ------------
```

This allows Player 2 to capture the central `(+)` piece since the <x> on the top left, the <+> on the top and the <+> on the right connect to that square.

```
+---+---+---+
|<x>|<+>|   |
+---+---+---+
|(x)|<+>|<+>|
+---+---+---+
|<+>|<x>|   |
 ------------
```

This allows Player 2 to then capture the `(x)` piece on the left

```
+---+---+---+
|<x>|<+>|   |
+---+---+---+
|<x>|<+>|<+>|
+---+---+---+
|<+>|<x>|   |
 -----------
```

Scoring

For each piece a player has put down, score 1 point. Additionally, each piece of a player is checked for the number of connections it has. If any of the squares a piece is connected to contains a piece belonging to the same player then the player scores an additional 1 point.

For examples, in the following diagram the <+> piece in the middle gains an additional 2 points because two of the squares it is connected to contains pieces belonging to that player.

```
+---+---+---+
|   |<+>|   |
+---+---+---+
|   |<+>|<x>|
+---+---+---+
|   |(+)|   |
-------------
```

In the following diagram, we can see the accumulated score for each player.

```
   -----------------------
0 |(+)|(+)|(+)|   |<+>|<x>|
  |---+---+---+---+---+---|
1 |(+)|(+)|(+)|   |<+>|<+>|
  |---+---+---+---+---+---|
2 |   |   |   |<x>|   |<x>|
  |---+---+---+---+---+---|
3 |<+>|   |   |   |   |(x)|
   -----------------------
    a   b   c   d   e   f
Player 1 score = 21 : Player 2 score = 17
```

Ending the Game

Players are limited to a maximum of 3 minutes to make all their moves

The game ends when one of the following happens

1. One of the players makes an illegal move.
2. One of the players has not finished all their moves in less than 3 minutes.
3. When there are no more empty spaces on the board.

Determining a Winner

If a player makes an illegal move or fails to make their moves within the allotted time then they lose and the other player is declared the winner. Otherwise, the winner is the player with the highest score. If the players have equal scores then the game is a draw.

Connections

In the /pup/prprog/assign1 directory I have placed a number of .cpp and .h files. I have also included a makefile. If you copy the files to your directory and run make it will compile the program to create an executable called connections. You and a friend can then run connections to play the game.

When you run the game, it will print a menu of instructions and then the board. You will notice that the board has the digits 0 to 3 on the left and the letters a to f below. These form the coordinates for any square on the board. Thus a0 is the top left square and f3 is the bottom right square. You move a piece into a square by specifying the coordinate of the square followed by a + or a x, depending on if you want to put a cross or a plus piece in the square.

If you examine the code you will see it uses concepts of polymorphism and inheritance to work. As this is not taught directly in the subject you will have to research how they operate in C++. The main thing is, all the rules of the game are implemented within the code. The only exception to this is the timing of the game.

timer
To keep track of the time while playing connections there is a program called timer. You can run timer thusly

```
/home/glingard/timer
```

Once you run the timer program it will wait till you enter s to start. Thereafter, just typing enter will get it to update the time for the current player. Each side alternates making their move on the connections program and then typing enter on the timer program. This will necessitate having a separate terminal session open for the connections program and the timer program.

The timer will automatically finish after each side has completed their moves. It will also end if you type q and then enter.

Requirement

You are to extend the connections code so the computer can play the game. That is, code the artificial intelligence (ai) so it can play like a human. To do this, you will have to modify the code so it can accept one of two command line arguments. If you type the following

```
connections -1
```

Your program will play Player 1 while you will play Player 1. If you type the following

```
connections -2
```

Your program will play Player 2 while you play Player 1.

Benchmark Program

To clarify how the program works, a benchmark executable version has been placed on the server. You can run it by typing the following command

```
    /home/glingard/connections [-1 | -2]
```

You can decide on which side the benchmark plays. To give you an idea of the size of this assignment, the addition to the connections code (including whitespace and comments) has taken about 100 lines of code.

You can automate running your ai against the benchmark by entering the following command

```
      /home/glingard/playgame [-1 | -2] connections
```

This assumes your ai executable is called connections. You may call it something else. You enter whether your executable is playing Player 1 or Player 2. Playgame will then run both the benchmark and your executable against each other, showing you how much time each is taking and keeping track of the score.

PLEASE NOTE. Your program must run in ***exactly*** the same way as the benchmark program.

Assignment Objectives

The purpose of this assignment is to demonstrate competence in the following skills.

- ❑ Algorithm design.
- ❑ Recursion.
- ❑ OO design.

These tasks reflect all the subject objectives apart from objective 4.

Queries

If you have a question, please contact the subject coordinator as soon as possible.

Gordon Lingard
glingard@it.uts.edu.au
9514-7935
Room 04.559, Building 10

However, for frequently asked questions a FAQ file will be put up. Please check this file before emailing the coordinator with a question.

```
      /pub/prprog/assign/assign1/faq.txt
```

If serious problems are discovered the class will be informed on UTS Online and an update will be included in the following file

```
      /pub/prprog/assign/assign1/errata.txt
```

Please keep a look out for this file.

PLEASE NOTE. If the answer to your questions can be found directly in any of the following
- ❑ subject outline
- ❑ assignment specification
- ❑ faq.txt
- ❑ errata.txt
- ❑ UTS Online discussion board
- ❑ DSA web page

You will be directed to these locations rather than given a direct answer.

PLEASE NOTE 2. Please do not send email in HTML format or with attachments. They will not be read or opened. Only emails sent in plain text format will be read. Emails without subject lines will be automatically deleted by the junk mail filters I have in place.

Assignment Submission

You will submit your program via the `submitass1` program. To do this you will need to zip all of your connection files. I should be able to unzip the file, run `make` and create the executable without any compiler errors or warnings.

You must start in the rerun directory that contains your .zip file and connections executable that is implementing your ai. You then run

```
    /home/glingard/submitass1 connections connections.zip
```

This assumes that you have zipped all your files to `connections.zip` and your executable is called `connections`.

Submission Lockout
The submission system will be set up to reject any submissions after the due date.

Plagiarism Agreement
You will be required to agree to a statement that the assignment is your own work and that you haven't given your code to anyone else. The only exception to this is other members in your group (see below).

If all goes well, `submitass1` will reply with a message saying you have successfully submitted the assignment. It will also place in your current directory a file called `receipt.txt`.

You may submit your assignment as many times as you like. The last assignment received will be the one marked.

PLEASE NOTE. Only assignments submitted via the `submitass1` program will be accepted for marking.

receipt.txt

`receipt.txt` is your proof that you have submitted your assignment. Once you have received it copy it to somewhere safe, such as your home directory. Additionally, copy your C++ file into the same location. Do not modify them in any way. If you wish to continue developing your program then do it on a duplicate file.

The `receipt.txt` file contains three pieces of information
1. A line saying you have submitted your file and when you did it.
2. A checksum of your zip file
3. A checksum of your receipt

If you tamper with your zip file or the receipt then the checksums will be become invalid and therefore your receipt will become invalid. No actions will be taken if receipts have invalid checksums.

Group Work

This assignment may be done individually or in groups of two. If you form a group then both students will receive the same mark.

If you decide to form a group you **MUST** email me the members of the group by the end of

Week 6 – 9/4/2011. I will respond with confirmation of the group. If I receive no email I will assume you are doing the assignment individually. This is an absolute time limit and I will accept no further emails regarding group composition after this date.

Once you have informed me of a group you cannot change it. If you have trouble with the operation of your group, ask the coordinator for advice (preferably ask as a group). If a member feels that the other is not contributing then arrange with the subject coordinator for the group to meet to seek a solution. In extreme cases the subject coordinator may permit the group to split into two individuals. Any complaints about the group must be brought to me before week 13. No complaints about group operation will be considered after week 12.

Acceptable Practice vs Academic Malpractice

- Do not let anyone submit their assignment from your account. The `submitass1` program copies your assignment into a secure directory based upon your user login name. Anyone else using your account will have <u>their</u> assignment placed in <u>your</u> directory. Students who do this will find themselves reported to the Faculty for possible academic malpractice and misuse of Faculty resources.
- Participants are reminded of the principles laid down in the "Statement of Good Practice and Ethics in Informal Assessment" in the Faculty Handbook. Assignments in this subject should be your own original work. Any collaboration with another participant should be limited to those matters described in the "Acceptable Behaviour" section. Any infringement by a participant will be considered a breach of discipline and will be dealt with in accordance with the Rules and By-Laws of the University. The Faculty penalty for serial misconduct of this nature is zero marks for the <u>subject</u>. For more information, see
  http://wiki.it.uts.edu.au/start/Academic_Integrity

Assignment Security

It is important to note that <u>you</u> have a responsibility to maintain the security of your assignment files. You can read more details about this in the plagiarism section of the DS & A web site - *http://learn.it.uts.edu.au/dsa/*

Assessment

Marks will be awarded out of 20. This will be scaled to 11% when calculating your final mark in the subject. The scheme for allocating marks is as follows:

Part 1 marks
Between 0 and 12 marks will be awarded based upon how well your compiled code runs against the benchmark program. This is automated and will be determined each time you run the submission program.

Your program will be run as Player 1 and then as Player 2 against the benchmark. For each game played, the following marks will be awarded.

0 Your code executes an illegal move or makes no moves at all.
2 Your program makes no illegal moves but times out before finishing all moves.
4 Your program finishes all moves in the allotted time but loses to the benchmark.
5 Your program finishes all moves in the allotted time and draws against the benchmark.
6 Your program finishes all moves in the allotted time and wins against the benchmark.

For example, if your code draws against the benchmark and then loses, you will get (5 + 4 = 9/12) marks.

You can submit your assignment as many times as you like, the best result you make against the benchmark is the one that will be used as your Part 1 mark.

PLEASE NOTE: Since the marking is based upon a competition, I reserve the right to make improvements to the implementation of the benchmark ai. The longer you delay submitting your ai implementation, the more likely the benchmark ai will be strengthened – thus making it more difficult to achieve a win against it.

Part 2 marks
Between 0 and 8 marks will be awarded as part of a competition held with other students in your laboratory for week 13.

These marks will ***ONLY*** be awarded if your program gets 8+ marks from Part 1. If your program has not attained this score then you will not be permitted to enter the student competition. Please examine the Part 1 scoring system carefully to see what is required to attain 8+ marks.

Details of the competition will be made available closer to the laboratory date.

PLEASE NOTE . Part 2 marks will not be awarded if you do not turn up to the laboratory.

Late Assignments, Extensions and Special Consideration

There will be **no** extensions for this assignment. If you fail to submit your assignment by the specified due date and time you will not be eligible to enter the competition in the week 13 laboratory.

Students may apply for special consideration if they consider that illness or misadventure has adversely affected their performance in the assignment. For more information go to

http://www.sau.uts.edu.au/exams_ass/spec_cons.html

Getting Marks

You can check on your marks by running the following program.

```
/home/glingard/getmarks
```

Apart from your marks this program will give the following information.

- Given out – the date the assignment is given out.
- Due date – the date the assignment is due.
- Marks released – The date the marks are released.
- Close date – The assignment is closed and the solution will be released. The close date will be one week after the assignments are marked and given back.

PLEASE NOTE. It is your responsibility to check that I have received your assignment and given you a mark. Even if you have a receipt you should check your mark and inform me if there is any problem before the close date. The `getmarks` program allows you to do this very easily. Unless you have a valid receipt or there are exceptional circumstances (e.g. serious medical conditions) no further correspondence regarding the assignment will be entered into after the close date.