Module CO32018 Internet Programming 3

Coursework for Even Semester May 2006 Napier University version

Aim

The aim of the coursework is to create an implementation of a puzzle on the WWW which can be manipulated and solved by the user.

Requirements

Attached to this document you will find a set of 10 puzzles of the sort that are traditionally set in Sunday papers and magazines. You are required to choose **ONE** of the puzzles described in this document and

1. create an implementation of it on your Web page so that it can be accessed via the Internet.
2. write a report about your implementation.

Deliverables of the Coursework

1. The Implementation

- The aim of the implementation is a graphical representation of the puzzle which provides the facilities that a player needs to be able to solve the problem: manipulating the symbols, appropriate description and explanation, and help where suitable. The program should make use of the visual programming facilities of the chosen representation to give the user an implementation of the puzzle which is friendly and easy to use. The code must be fully tested and must be able to be demonstrated on any reasonable platform as defined below.
- This coursework is intended to give you an opportunity to demonstrate that you have acquired skills covered in the lecture material. It is therefore essential that a substantial amount of the programming content of the submission is in one of the free or open source/public domain languages covered by the lectures. A list of these is given in the appendix to this introduction. Marks will be deducted for a submission which makes excessive use of proprietary software such as Flash and ASP.
- Your implementation must be viewable via the WWW on any reasonable Internet enabled computer system: for example, a PC running Linux or Windows XP , using a popular browser such as Mozilla Firefox, Internet Explorer or Opera. You may assume that Java and Javascript are enabled on the Web browser used to view the program. It is your own responsibility to ensure that your program will perform satisfactorily on any reasonable platform. You should note that Java Swing components will often not be viewable when downloaded from the Internet via a URL.

- The top level HTML file containing your code **must** be named **co32018.html** and it **must** be made available as a WWW publication to the lecturing staff and the class, by being placed in a public_html directory on a web server with appropriate permissions set.

- Please note the program is not required to be able to solve the puzzle or to be able to recognise that a solution has been found. Credit will however be given for including these as "extensions" (see below) where appropriate.

- Credit will be given for the inclusion of appropriate extensions to the puzzle. Extended implementations might be able to perform one or more of the following operations demonstrate a solution to the puzzle:

    - recognise that the user has solved the problem.
    - offer alternative solutions.
    - The assignment is open-ended, and credit will be given for extensions that are appropriate to the problem you have chosen.
- Credit will also be given to originality of implementation and also for adventurousness in choosing the problem.

- There will also be credit for designing your solution in such a way that it performs gracefully if an attempt is made to run it on a platform which does not provide all the facilities for which it was designed:- for example, if the program was designed to use frames, but a user with an older version of a browser attempts to run the program without the use of frames.

2. Report on the program

The report on the program should be no more than 10 pages (excluding code) and is to include the following:

- a description of the problem and an account of your solution appropriate to the implementation chosen (for example, if you have used Java, your account should include brief descriptions of the classes and objects you have defined, and any special facilities of the language that you have found useful).
- an account of any sources used or adapted in building the program. You are free to look at Web sources for inspiration and ideas in arriving at your programs. However, all such sources must be credited. Marks will be deducted if the submission does not contain sufficient original content.
- a brief critical appraisal of the program.
- a printout of the code of the program.
- screenshots of your program to demonstrate that it will run successfully on two distinct browsers: e.g. Firefox, Internet Explorer, Opera. Each screenshot **must** include a URL that shows that the program has been accessed from a website and not from a local file.

- documentation appropriate to the programming language used. For example, for Java you should make use of Javadoc. For other languages, documentation should be provided to a similar standard. Javadoc or other documentation may be submitted as an HTML document linked to the submission. There must be a clear statement in the body of the report to indicate where the documentation is to be found.

Conditions of Acceptance

Please read the following conditions carefully. Failure to observe any of these conditions will result in loss of marks:

1. The program MUST be submitted in a World Wide Web accessible form

As marker and moderator it is absolutely essential that I can view your program at a URL via a standard browser. As a Napier University student you are provided with a public_html folder which permits you to publish a web page using your userid as part of the URL. You can therefore publish your program with a URL with the general form:

www.soc.napier.ac.uk/-userid/co32018.html (or co32018.htm)

If you are studying at Napier University you MUST publish your program with a URL of this form..

If you are studying at one of the franchised colleges of Universities, you may use a URL provided by your institution provided that the URL is viewable externally.

If your institution does not provide an externally viewable URL, you may publish your program on the Napier University web server. Your files can be transferred to this server by FTP; full instructions for this are given on the Internet Programming web pages.

2. The puzzle represented MUST be one of those described in this document

You are not allowed to choose a puzzle that is not in the list, and your program must be a reasonable representation of the requirements of one of the puzzles.

3. The top-level file of the puzzle implementation MUST be named co32018.html (or co32018.htm).

4. The front page of the report MUST clearly state the URL where the implementation is to be found.

5. All supplementary material submitted must be clearly labelled

You may if you wish submit backup copies of your work on CD or floppy disk. Such material will not be looked at unless there is a technical problem with the delivery of the material via the WWW which you could not have been expected to have anticipated.

Any such backup material **must be** properly labelled with your name and Napier University matriculation number. Unlabelled media will be discarded.

6. The work submitted must be demonstrably your own

You are of course free to research sources, including internet sources, for material appropriate to include in your submission – either the program or the report. However, all souch sources must be clearly credited. **Uncredited use or excessive dependence on material downloaded from the Internet will be regarded as plagiarism and dealt with as such.**

Submission Date

The report must be handed into the Courses Office not later than 4.00 pm on Thursday 4th May. The program must be put in place in your public_html folder by the same date.

Submission

The top level HTML file containing your code must be named **co32018.html** or **co32018.htm** and it must be made available as a WWW publication to the lecturing staff and the class, by being placed in a public_html directory on a web server with appropriate permissions set.

Assessment

The assessment process may include a peer assessment exercise in which, working in small groups, you will compare the quality of the programs produced by the members of the class. **Note that this exercise will not necessarily be carried out on the same platform as that used for the ordinary tutorials.**

Module Leader:

L. Morss

School of Computing
Napier University

Feb 20, 2006

Marking Scheme

Report on the Program	**60%**	Description of the Problem Sources used: Critical Appraisal: Documented Code: Appropriate Screen Shots: **Total: the report total is multiplied by 0.6 to give a mark for the report out of 60%**	15/100 15/100 25/100 30/100 15/100
Extensions	**10%**		
Program Quality	**30%**	Where a peer assessment exercise has taken place: **5%** for attendance at the exercise **25%** assigned by peers	

Appendix - Programming and Scripting Languages

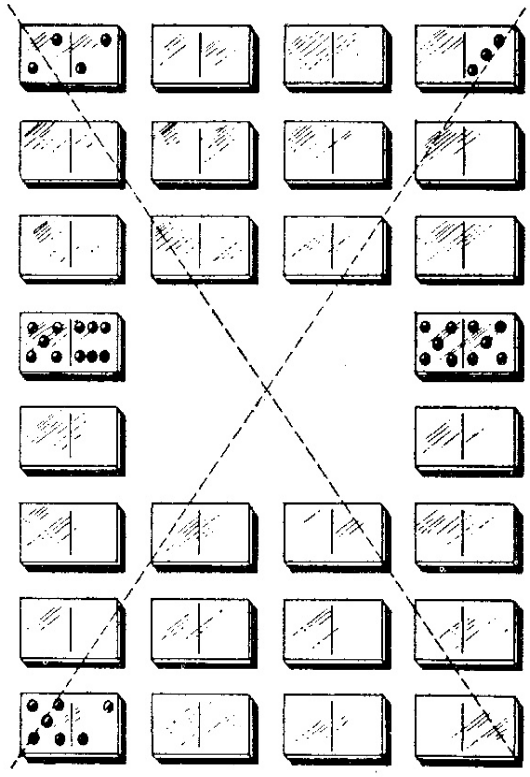
The programming and scripting languages covered in the lecture material of the module are listed below. A substantial amount of the programming content of the submission must be in one of these languages:

Java, Javascript, Perl, PHP

It is expected that HTML or XML (with extensions such as CSS) will be the markup language in which the program is embedded.

The Puzzles

1. A Magic Square with a Hole (MP#204)



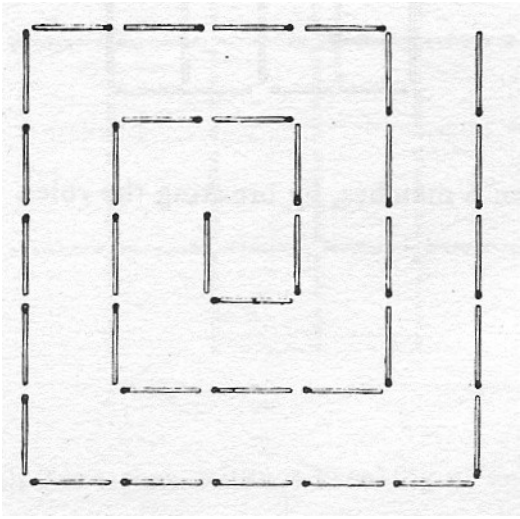
"Using all 28 dominoes, form a rectangular array with a central hole, as shown. The sum of each of the eight rows, eight columns, and each of the diagonals (shown by by dotted lines) must be 21. For rows, add whole dominoes; for columns, add half dominoes (squares) and for diagonals, add the 6 half-dominoes covered by the dotted line.

"The fourth row from the top has been filled in. Its sum is 5+6+5+5=21. The four corner pieces have also been filled in, including the double blank in the bottom right corner."

Your implementation should be able to keep track of the sums of all the rows, diagonals and columns, and be able to allow the user to move dominoes from a stockpile into the allowed positions in the layout.

2. A Spiral with Matches (MP#125)

A figure resembling a spiral is shown with 35 matches. Move 4 matches to form 3 squares.

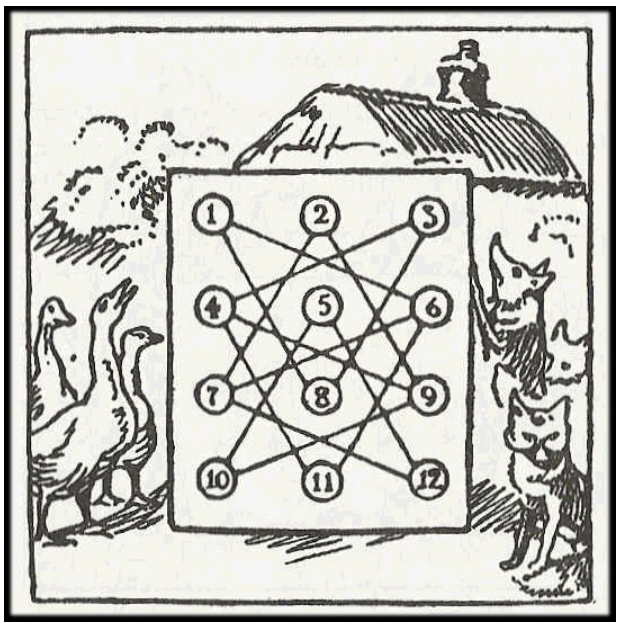


3. Exploring the Desert (CIP#294)

"Nine travellers, each possessing a motor-car, meet on the eastern edge of a desert. They wish to explore the interior, always going west. Each car can travel 40 miles on the contents of a tank, which holds a gallon of petrol, and each can carry nine extra gallon tins of petrol and no more. Only unopened tins can be transferred from car to car. What is the greated distance to which they can enter the desert without making any depots for petrol for the return journey?"

This puzzle should lend itself well to a graphical representation.

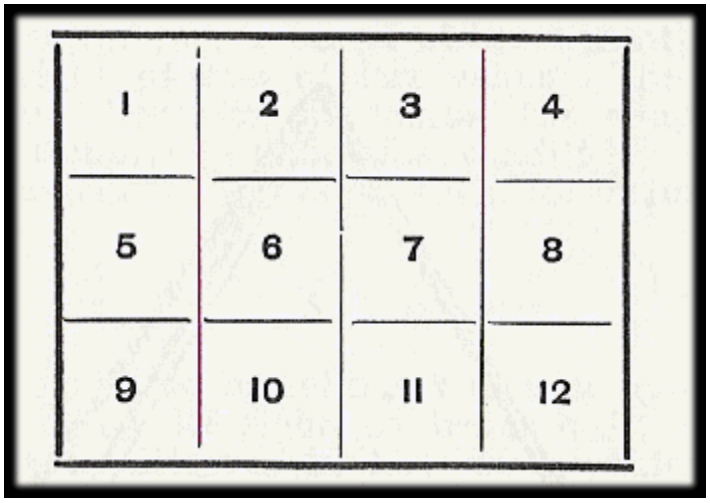
4. Foxes and Geese Canterbury Puzzles #94



The figure shows the layout for a little puzzle called “foxes and geese.” Three counters marked to represent foxes are placed on circles numbered 10, 11, and 12; and three counters marked to represent geese are placed on circles numbered 1,2 , and 3. You have to make the foxes and geese change places by moving their counters alternately from one circle to a neighbouring one. But you must do so without ever allowing a fox to get into a circle which neighbours a circle that contains a goose.

Your program should keep count of the number of moves made, and hence allow you find out how to solve the puzzle in the least number of moves.

5. Counting the Postage Stamps AIM #285



“Suppose you have just bought twelve postage stamps in this form – three by four – and a friend asks you to oblige him with four stamps, all joined together – no stamp hanging on by a mere corner. In how many ways is it possible to tear off those four stamps? You see, you can give him 1,2,3,4, or 2,3,6,7, or 1,2,3,6 and so on. Can you get the exact number?”

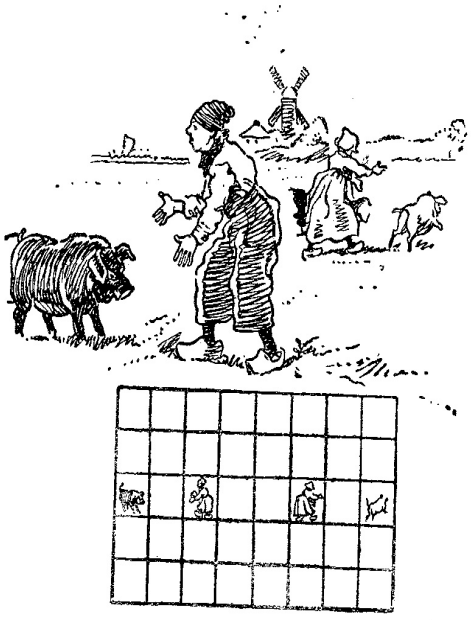
The actual number is fairly large (> 50) so the program should keep track of the combinations that have already been chosen – so that the player can't pick a combination twice – and should also confirm that a selection is legal according to the rules of the puzzle.

6. Queen Moves (CIP#277)

The problem is to place the Queen on a chessboard and pass her over the entire 64 squares and back again to the point of beginning in fourteen moves.

The Queen must make normal moves – vertical, horizontal or diagonal – and the implementation should highlight the squares that have already been covered.

7. Catching the Hogs (CIP#284)



“In the illustration, Hendrick and Katrun are seen engaged in the exhilerating sport of attempting the capture of a couple of hogs. Why did they fail?

The Dutchman and his wife are represented as if they are coloured counters on squared paper as shown in the picture. The player moves the Dutchman and his wife each one square either left or right or up or down – but not diagonally. The computer (or a second player) now moves both pigs one square either left or right or up or down – but not diagonally. The game continues in this way until Hendrick has caught one hog and Katrun the other.

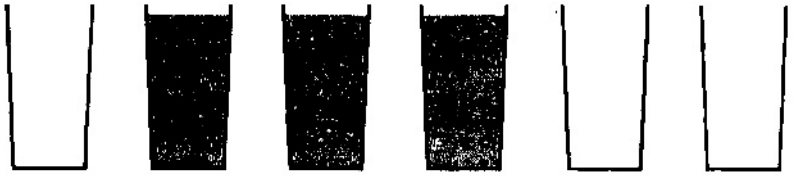
It should be possible to produce an implementation that allows the user to work out a strategy for catching the pigs.

8. The Golf Puzzle (SL#77)

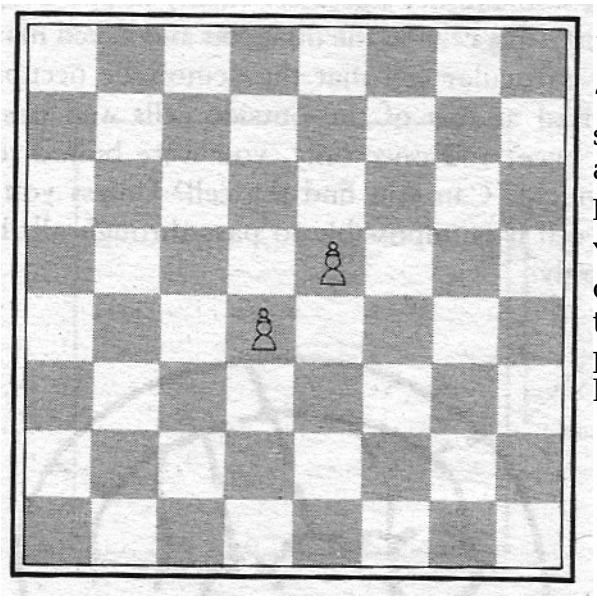
A genius has invented a winning system for playing golf based on two strokes: a drive and an approach. What should be the proper length of these two strokes to make possible the lowest score on a nine-hole course, given that the lengths of the holes are 150 yards, 300 yards, 250 yards, 325 yards, 275 yards, 350 yards, 225 yards, 400 yards, and 425 yards?

Each stroke must be played directly towards the hole, and the ball must go the full length on each stroke, but you may go beyond the hole in either stroke then play back toward the hole. Both the drive and the approach is an exact multiple of 25 yards.

9. Tricky Tumblers CIP #446



There are 6 pint glasses, three full and three empty arranged as shown. Write a program which allows the player to pick up any of the glasses and place it either at the end of the row or else in the space left from where a glass has previously been moved. The object is to arrange the glasses so that they are alternately full and empty, using the minimum number of moves. The program could keep track of the moves taken and allow the user to replay the best game "so far".



10.A Puzzle with Pawns (CIP#295)

“Place two pawns on the middle of a chessboard as shown. Now place the remaining 14 pawns (16 in all) so that no three shall be in a straight line in any possible direction. “

Your implementation could highlight the rows, columns and diagonals which have two pawns on them at any stage of game. so that the remaining pawns can only be placed on squares that are not highlighted.

Sources:

MP - Moscow Puzzles B. A. Kordemsky Penguin (1976)

AIM - Amusements in Mathematics H. E. Dudeney Dover (1970)

CP – Canterbury Puzzles H. E. Dudeney Dover (1986)

SL – Mathematical Puzzles of Sam Loyd - Martin Gardner Dover (1959)

CIP – Curious and Interesting Puzzles - David Wells Penguin (1992)