AQA

General Certificate of Education
Advanced Subsidiary Examination
June 2010

Computing   COMP1/PM/PY2

Unit 1   Problem Solving, Programming, Data Representation and Practical Exercise

Preliminary Material

A copy of this Preliminary Material will appear on the GCE Computing front page on the AQA Website on Monday 1 March 2010. It will include the Skeleton Program as issued by AQA.

A copy of the Preliminary Material, including any amendments to the Skeleton Program made by the Head of Computing and approved by AQA, must be given to candidates by their centre on or after Thursday 1 April 2010.

Information for Candidates

● This Preliminary Material comprises
       – Instructions to Candidates and a
       – Skeleton Program for **Python 2.5** which your teacher will supply.
  You must **only** use the version of the Skeleton Program supplied by your teacher.

● Candidates are advised to familiarise themselves with the Preliminary Material before the examination.

● You must **not** take any copy of the Preliminary Material or any other material into the examination room. A second copy of the Preliminary Material will be given to you in the examination. Your teacher will also provide you with access electronically to the Skeleton Program at the start of the examination.

         COMP1/PM/PY2

Instructions to Candidates

The question paper is divided into four sections and a recommendation is given to candidates as to how long to spend on each section. Below are the recommended timings for the 2010 examination.

Section A
You are advised to spend no more than **35 minutes** on this section.
Questions will examine the specification content **not** specific to the *Preliminary Material*.

Section B
You are advised to spend no more than **20 minutes** on this section.
You will be asked to create a new program **not** related to the *Preliminary Material* or Skeleton Program.

Section C
You are advised to spend no more than **15 minutes** on this section.
Questions will refer to the *Preliminary Material* and the Skeleton Program, but will not require programming.

Section D
You are advised to spend no more than **50 minutes** on this section.
Questions will use the Skeleton Program and the *Preliminary Material*.

Electronic Answer Document

Answers to all questions must be written into the word processed document made available to you at the start of the examination and referred to in the question paper as the Electronic Answer Document.

Preparation for the Examination

For your programming language you should ensure that you are familiar with this *Preliminary Material*, including the Skeleton Program.
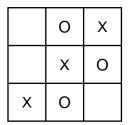
'Noughts and Crosses Game'

The **Skeleton Program** in this Preliminary Material is for a program based on the two-player game of 'Noughts and Crosses'.

When playing Noughts and Crosses players take turns to place a symbol on the board. The board is a 3x3 grid, which is initially empty. One player uses the symbol 'X', the other the symbol 'O'. Players are only allowed to place a symbol in an empty position on the board. The aim of the game is to be the first person to get three of their symbols in a line – a line of symbols can be either in a row, in a column or along a diagonal. Three winning board positions for the player using the symbol 'X' are shown in **Figure 1**.
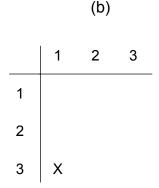
Figure 1

X	X	X
	O	
	O	

O	X	
O	X	O
	X	

	O	X
	X	O
X	O	

In the **Skeleton Program** players use row and column coordinates to specify where they wish their 'X' or 'O' symbol to be placed. First, they specify the x coordinate – this indicates which column they want; then they specify the y coordinate – this indicates which row they want. **Figure 2** shows part of a game. The first board, (a), is the initial position, the second board, (b), shows what would happen if the 'X' player enters coordinates of (1,3).

Figure 2

| (a) | | | (b) | | |

```
      1   2   3              1   2   3
   ┌─────────────        ┌─────────────
 1 │                   1 │
 2 │                   2 │
 3 │                   3 │ X
```

The game finishes when a player gets three of their symbol in a line or there are no empty cells on the grid. If the grid is full but neither player has a line of three symbols then the game is drawn. After the last move the result of the game is displayed.

Outline Design

The Structured English description of an algorithm for playing one game is as follows.

```
SET number of moves TO 0
Clear Board
Display Board
IF starting symbol IS EQUAL TO player one's symbol
   THEN OUTPUT message saying player one is going to make a move first
   ELSE OUTPUT message saying player two is going to make a move first
ENDIF
SET current symbol TO starting symbol
DO
   DO
     Get move coordinates
     Check if move is valid
     IF NOT valid move
       THEN OUTPUT invalid move message
     ENDIF
   UNTIL valid move
   Make move
   Display Board
   Check if game has been won
   INCREMENT number of moves BY 1
   IF NOT game has been won
     THEN
       IF number of moves IS EQUAL TO 9
         THEN game has been drawn
         ELSE change current symbol
       ENDIF
   ENDIF
UNTIL game has been won OR game has been drawn
OUTPUT result of game
```

Variables

The main variables used are as follows.

Identifier	Data Type	Purpose
NoOfMoves	Integer	Used to keep track of how many moves have been made in the game so far ie how many positions in the board have a symbol in them
CurrentSymbol	Char	Used to indicate the symbol being used by the player whose turn it currently is - 'X' or 'O'
Board	Array[1..3,1..3] Of Char	A two-dimensional array used to indicate which symbol, if any, is currently in each position on the 3x3 grid.

Notes

The programming language used to code the game will determine the letter case for each identifier, and so may not match exactly the identifiers shown in the table above.

Your chosen programming language may use arrays with a lower bound value of 0. If so, array cells with indices of 0 are not used.

END OF INSTRUCTIONS TO CANDIDATES

```
# Skeleton Program Code for AQA Comp1 Summer 2010 Examination
# this code should be used in conjunction with the Preliminary
# Materials
# written by the AQA COMP1 Programmer Team
# developed using Python v2.5

import random

Board = [[0,0,0,0],
         [0,0,0,0],
         [0,0,0,0],
         [0,0,0,0],
         ]
PlayerOneName = ""
PlayerTwoName = ""
XCoord = 0
YCoord = 0
NoOfMoves = 0
ValidMove = False
GameHasBeenWon = False
GameHasBeenDrawn = False
CurrentSymbol = ""
StartSymbol = ""
PlayerOneSymbol = ""
PlayerTwoSymbol = ""
Answer = ""
PlayerOneScore = 0.0
PlayerTwoScore = 0.0

def DisplayBoard(Board):
    print "  | 1 2 3 "
    print "--+-------"
    for Row in range(1,4):
```

```python
        print str(Row) + " |",
        for Column in range(1,4):
            print Board[Column][Row],
        print
    print "\n"

def ClearBoard(Board):
    for Row in range(1,4):
        for Column in range(1,4):
            Board[Column][Row] = " "

def GetMoveCoordinates():
    X = int(raw_input("Enter x coordinate: "))
    Y = int(raw_input("Enter y coordinate: "))
    return X, Y

def CheckValidMove(XCoordinate, YCoordinate, Board):
    ValidMove = True
    # Check x coordinate is valid
    if (XCoordinate < 1) or (XCoordinate > 3):
        ValidMove = False
    return ValidMove

def CheckXOr0HasWon(Board):
    XOrOHasWon = False
    for Column in range(1,4):
        if (Board[Column][1] == Board[Column][2]) and (Board[Column][2] == Board[Column][3]) and (Board[Column][2] != " "):
            XOrOHasWon = True
    for Row in range(1,4):
        if (Board[1][Row] == Board[2][Row]) and (Board[2][Row] == Board[3][Row]) and (Board[2][Row] != " "):
            XOrOHasWon = True
    return XOrOHasWon
```

```python
def GetWhoStarts():
    RandomNo = random.randint(0, 100)
    if (RandomNo % 2) == 0:
        WhoStarts = "X"
    else:
        WhoStarts = "O"
    return WhoStarts


if __name__ == "__main__":
    PlayerOneName = raw_input("What is the name of player one? ")
    PlayerTwoName = raw_input("What is the name of player two? ")
    while PlayerOneSymbol not in ["X","O"]:
    # Choose player one's symbol
        PlayerOneSymbol = raw_input(PlayerOneName + " what symbol do you wish to use, X or O? ")
        if PlayerOneSymbol not in ["X","O"]:
            print "Symbol to play must be uppercase X or O"
    if PlayerOneSymbol == "X":
        PlayerTwoSymbol = "O"
    else:
        PlayerTwoSymbol = "X"
    StartSymbol = GetWhoStarts()
    while Answer not in ["N","n"]:  # Play a game
        NoOfMoves = 0;
        GameHasBeenDrawn = False;
        GameHasBeenWon = False;
        ClearBoard(Board);
        print "\n"
        DisplayBoard(Board);
        if StartSymbol == PlayerOneSymbol:
            print PlayerOneName + " starts playing " + StartSymbol
        else:
            print PlayerTwoName + " starts playing " + StartSymbol
        print "\n"
```

```
CurrentSymbol = StartSymbol
while (not GameHasBeenWon) and (not GameHasBeenDrawn): # Play until a player wins or the game is drawn
    ValidMove = False;
    while not ValidMove: # Get a valid move
        XCoord,YCoord = GetMoveCoordinates()
        ValidMove = CheckValidMove(XCoord, YCoord, Board)
        if not ValidMove:
            print "Coordinates invalid, please try again"
    Board[XCoord][YCoord] = CurrentSymbol
    DisplayBoard(Board)
    GameHasBeenWon = CheckXOr0HasWon(Board)
    NoOfMoves += 1
    if not GameHasBeenWon:
        if NoOfMoves == 9: # Check if maximum number of allowed moves has been reached
            GameHasBeenDrawn = True
        else:
            if CurrentSymbol == "X":
                CurrentSymbol = "O"
            else:
                CurrentSymbol = "X"
if GameHasBeenWon:   # Update scores and display result
    if (PlayerOneSymbol == CurrentSymbol):
        print PlayerOneName + " congratulations you win!"
        PlayerOneScore += 1
    else:
        print PlayerTwoName + " congratulations you win!"
        PlayerTwoScore += 1
else:
    print "A draw this time!"
print "\n"
print PlayerOneName + ", your score is: " + str(PlayerOneScore)
print PlayerTwoName + ", your score is: " + str(PlayerTwoScore)
print "\n"
```

```python
if (StartSymbol == PlayerOneSymbol):
    StartSymbol = PlayerTwoSymbol
else:
    StartSymbol = PlayerOneSymbol;
Answer = raw_input("Another game Y/N? ")
```