



General Certificate of Education
Advanced Subsidiary Examination
June 2010

Computing

COMP1/PM/JA

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 **Java** 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.

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)
1 2 3
1
2
3

(b)
1 2 3
1
2
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.

Turn over ►

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

```
package compl_java_2010;

import java.util.Random;

/* Skeleton program code for the AQA Comp1 Summer 2010 examination
 * this code should be used in conjunction with the Preliminary
 * Material, written by the AQA COMP1 Programmer Team developed in Netbeans 6.7
 * Note the private modified has been left out to make the program
 * easier to read for AS students. The AQA Console class is used.
 */

public class Main {

    // class Coordinate facilitates the passing of
    // information back through a parameter
    class Coordinate {
        int value;
    } // end inner class Coordinate
    char board[][] = new char[4][4];
    String playerOneName;
    String playerTwoName;
    float playerOneScore;
    float playerTwoScore;
    int xCoord;
    int yCoord;
    boolean validMove;
    int noOfMoves;
    boolean gameHasBeenWon;
    boolean gameHasBeenDrawn;
    char currentSymbol;
    char startSymbol;
    char playerOneSymbol;
    char playerTwoSymbol;
    char answer;
    //Console is specifically for the Java version
    Console console = new Console();

    public Main() {
        playerOneName = console.readLine("What is the name of player one? ");
        playerTwoName = console.readLine("What is the name of player two? ");
    }
}
```

```

playerOneScore = 0;
playerTwoScore = 0;
do { //Choose player one's symbol
    playerOneSymbol = console.readChar((playerOneName +
        " what symbol do you wish to use X or O? "));
    if (playerOneSymbol != 'X' && playerOneSymbol != 'O') {
        console.println("Symbol to play must be uppercase X or O");
    } // end if
} while ((playerOneSymbol != 'X' && playerOneSymbol != 'O'));
if (playerOneSymbol == 'X') {
    playerTwoSymbol = 'O';
} else {
    playerTwoSymbol = 'X';
} // end if/else
startSymbol = getWhoStarts();
do { // Play a game
    noOfMoves = 0;
    gameHasBeenDrawn = false;
    gameHasBeenWon = false;
    clearBoard(board);
    console.println();
    displayBoard(board);
    if (startSymbol == playerOneSymbol) {
        console.println(playerOneName + " starts playing " + startSymbol);
    } else {
        console.println(playerTwoName + " starts playing " + startSymbol);
    } // end if/else
    console.println();
    currentSymbol = startSymbol;
    do { //Play until a player wins or the game is drawn
        do { //Get a valid move
            Coordinate x = new Coordinate();
            Coordinate y = new Coordinate();
            getMoveCoordinates(x, y);
            xCoord = x.value;
            yCoord = y.value;
            validMove = checkValidMove(xCoord, yCoord, board);
            if (!validMove) {
                console.println("Coordinates invalid, please try again");
            } // end if
        } while (!validMove);
    }
}

```

```

board[xCoord][yCoord] = currentSymbol;
displayBoard(board);
gameHasBeenWon = checkXOrOHasWon(board);
noOfMoves++;
if (!gameHasBeenWon) {
    // Check if maximum number of
    // allowed moves has been reached
    if (noOfMoves == 9) {
        gameHasBeenDrawn = true;
    } else {
        if (currentSymbol == 'X') {
            currentSymbol = 'O';
        } else {
            currentSymbol = 'X';
        } // end if/else
    } // end if/else
} // end if
} while (!gameHasBeenWon && !gameHasBeenDrawn);
// Update scores and display result
if (gameHasBeenWon) {
    if (playerOneSymbol == currentSymbol) {
        console.println(playerOneName + " congratulations you win!");
        playerOneScore++;
    } else {
        console.println(playerTwoName + " congratulations you win!");
        playerTwoScore++;
    } // end if/else
} else {
    console.println("A draw this time!");
} // end if/else
console.println();
console.println(playerOneName + " your score is: " + String.valueOf(playerOneScore));
console.println(playerTwoName + " your score is: " + String.valueOf(playerTwoScore));
console.println();
if (startSymbol == playerOneSymbol) {
    startSymbol = playerTwoSymbol;
} else {
    startSymbol = playerOneSymbol;
} // end if/else
answer = console.readChar("Another game Y/N? ");
} while (answer != 'N' && answer != 'n');

```



```

} // end constructor Main

void displayBoard(char[][] board) {
    int row;
    int column;
    console.println(" | 1 2 3 ");
    console.println("--+-----");
    for (row = 1; row <= 3; row++) {
        console.write(row + " | ");
        for (column = 1; column <= 3; column++) {
            console.write(board[column][row] + " ");
        } // end column
        console.println();
    } // end row
} // end method displayBoard

void clearBoard(char[][] board) {
    int row;
    int column;
    for (row = 1; row <= 3; row++) {
        for (column = 1; column <= 3; column++) {
            board[column][row] = ' ';
        } // end column
    } // end row
} // end method clearBoard

void getMoveCoordinates(Coordinate xCoordinate, Coordinate yCoordinate) {
    xCoordinate.value = console.readInteger("Enter x Coordinate: ");
    yCoordinate.value = console.readInteger("Enter y Coordinate: ");
} // end method getMoveCoordinates

boolean checkValidMove(int xCoordinate, int yCoordinate, char[][] board) {
    boolean validMove;
    validMove = true;
    //Check the x Coordinate is valid
    if (xCoordinate < 1 || xCoordinate > 3) {
        validMove = false;
    } // end if
    return validMove;
} // end method checkValidMove

```

```

boolean checkXOrOHasWon(char[][] board) {
    boolean xOrOHasWon;
    int row;
    int column;
    xOrOHasWon = false;
    for (column = 1; column <= 3; column++) {
        if (board[column][1] == board[column][2] &&
            board[column][2] == board[column][3] &&
            board[column][2] != ' ') {
            xOrOHasWon = true;
        } // end if
    } // end column
    for (row = 1; row <= 3; row++) {
        if (board[1][row] == board[2][row] &&
            board[2][row] == board[3][row] &&
            board[2][row] != ' ') {
            xOrOHasWon = true;
        } // end if
    } // end row
    return xOrOHasWon;
} // end method checkXOrOHasWon

char getWhoStarts() {
    char whoStarts;
    int randomNo;
    Random objRandom = new Random();
    randomNo = objRandom.nextInt(100);
    if (randomNo % 2 == 0) {
        whoStarts = 'X';
    } else {
        whoStarts = 'O';
    } // end if/else
    return whoStarts;
} //end method getWhoStarts

public static void main(String[] args) {
    new Main();
} //end method main
} // end class main

```

```
package compl_java_2010;

import java.io.BufferedReader;
import java.io.FileReader;
import java.io.IOException;
import java.io.InputStreamReader;

/* Class for use in AQA COMP1 exams for students using Java.
 *
 * class Console will be made available for students to use,
 * although they are not restricted to just using it for i/o to the console.
 * If they know how to use java.io.Console that is fine.
 * Also, System.out.println can be used for outputs.
 * This class should save time in the exam, especially if they
 * have been used by students beforehand.
 */
class Console {

    public Console() {
        } // end of constructor CommandLineInput
        //returns the first character of a line typed in.

    public char readChar() {
        return readChar("");
    } // end method readChar

    public char readChar(String prompt) {
        return readLine(prompt).charAt(0);
    } // end method readChar

    public byte readByte(String prompt) {
        try {
            return Byte.parseByte(readLine(prompt).substring(0, 1));
        } catch (NumberFormatException nfe) {
            println(nfe.toString() + "Parsing a Byte");
        } catch (Exception e) {
            println("another exception" + e.toString());
        }
        return -1;
    }
}
```

```

} // end method readByte

public byte readByte() {
    return readByte("");
} // end method readByte

public String readLine() {
    return readLine("");
} // end method readLine with no parameter

public String readLine(String prompt) {
    String input = "";
    System.out.print(prompt);
    InputStreamReader isr = new InputStreamReader(System.in);
    BufferedReader br = new BufferedReader(isr);
    try {
        input = br.readLine();
    } catch (IOException ioe) {
        println("IO Error reading from command line.");
        System.exit(1);
    }
    return input;
} // end method readLine

public int readInteger(String prompt) {
    try {
        return (int) Integer.parseInt(readLine(prompt));
    } catch (NumberFormatException nfe) {
        System.out.println(" Number format excpetion.");
        System.out.println(nfe.toString());
    } // end try/catch
    return -1;
} // end method readInteger

public void print(Object o) {
    String output = String.valueOf(o);
    System.out.print(output);
} // end method print

public void println() {
    println("");
}

```

```
} // end method println with no parameter

public void println(Object o) {
    String output = String.valueOf(o);
    System.out.println(output);
} // end method println

public void write(Object o) {
    print(o);
} // end method write

public void writeLine() {
    println();
} // end method writeLine with no parameter

public void writeLine(Object o) {
    println(o);
} // end method writeLine

public void printf(String format, Object args) {
    System.out.printf(format, args);
} // end method printf
} // end class Console
```