SPECIMEN MATERIAL

AQA

AS
COMPUTER SCIENCE
(7516/1A)

Skeleton Program

```
// Skeleton Program code for the AQA AS Unit 1 SAM
// this code should be used in conjunction with the Preliminary Material
// written by the AQA COMP1 Programmer Team
// developed in the Visual C# 2008 Express programming environment

// Version 1.1

using System;
using System.IO;

namespace CSPreASSkelton
{
    class Program
    {
        public const int NO_OF_TRAPS = 2;
        public const int N_S_DISTANCE = 5;
        public const int W_E_DISTANCE = 7;

        public struct CellReference
        {
            public int NoOfCellsSouth;
            public int NoOfCellsEast;
        }

        static void Main(string[] args)
        {
            char[,] Cavern = new char[N_S_DISTANCE, W_E_DISTANCE];
            int Choice = 0;
            Boolean MonsterAwake = false;
            CellReference FlaskPosition = new CellReference();
            CellReference MonsterPosition = new CellReference();
            CellReference PlayerPosition = new CellReference();
            CellReference[] TrapPositions = new CellReference[NO_OF_TRAPS];
            while (Choice != 9)
            {
                DisplayMenu();
                Choice = GetMainMenuChoice();
                switch (Choice)
                {
                    case 1:
                        SetUpGame(Cavern, TrapPositions, ref MonsterPosition, ref
PlayerPosition, ref FlaskPosition, ref MonsterAwake, true);
                        PlayGame(Cavern, TrapPositions, ref MonsterPosition, ref
PlayerPosition, ref FlaskPosition, ref MonsterAwake);
                        break;
                    case 2:
                        LoadGame(TrapPositions, ref MonsterPosition, ref
PlayerPosition, ref FlaskPosition, ref MonsterAwake);
                        SetUpGame(Cavern, TrapPositions, ref MonsterPosition, ref
PlayerPosition, ref FlaskPosition, ref MonsterAwake, false);
                        PlayGame(Cavern, TrapPositions, ref MonsterPosition, ref
PlayerPosition, ref FlaskPosition, ref MonsterAwake);
                        break;
```

```
                    case 3:
                        SaveGame(TrapPositions, MonsterPosition, PlayerPosition,
FlaskPosition, MonsterAwake);
                        break;
                    case 4:
                        SetUpTrainingGame(Cavern, TrapPositions, ref
MonsterPosition, ref PlayerPosition, ref FlaskPosition, ref MonsterAwake);
                        PlayGame(Cavern, TrapPositions, ref MonsterPosition, ref
PlayerPosition, ref FlaskPosition, ref MonsterAwake);
                        break;
                }
            }
        }

        public static void DisplayMenu()
        {
            Console.WriteLine("MAIN MENU");
            Console.WriteLine();
            Console.WriteLine("1.  Start new game");
            Console.WriteLine("2.  Load game");
            Console.WriteLine("3.  Save game");
            Console.WriteLine("4.  Play training game");
            Console.WriteLine("9.  Quit");
            Console.WriteLine();
            Console.WriteLine("Please enter your choice: ");
        }

        public static int GetMainMenuChoice()
        {
            int Choice;
            Choice = int.Parse(Console.ReadLine());
            Console.WriteLine();
            return Choice;
        }

        public static void ResetCavern(char[,] Cavern)
        {
            int Count1;
            int Count2;
            for (Count1 = 0; Count1 < N_S_DISTANCE; Count1++)
            {
                for (Count2 = 0; Count2 < W_E_DISTANCE; Count2++)
                {
                    Cavern[Count1, Count2] = ' ';
                }
            }
        }

        public static CellReference GetNewRandomPosition()
        {
            CellReference Position = new CellReference();
            Random rnd = new Random();
            do
            {
```

```
                // a random coordinate of (0,0) needs to be rejected as this is
the starting position of the player
                Position.NoOfCellsSouth = rnd.Next(0, N_S_DISTANCE);
                Position.NoOfCellsEast = rnd.Next(0, W_E_DISTANCE);
            } while (Position.NoOfCellsSouth == 0 && Position.NoOfCellsEast == 0);
            return Position;
        }

        public static void SetPositionOfItem(char[,] Cavern, ref CellReference
ObjectPosition, char Item, Boolean NewGame)
        {
            CellReference Position = new CellReference();
            if (NewGame && Item != '*')
            {
                do
                {
                    Position = GetNewRandomPosition();
                } while (Cavern[Position.NoOfCellsSouth, Position.NoOfCellsEast]
!= ' ');
                ObjectPosition = Position;
            }
            Cavern[ObjectPosition.NoOfCellsSouth, ObjectPosition.NoOfCellsEast] =
Item;
        }

        public static void SetUpGame(char[,] Cavern, CellReference[]
TrapPositions, ref CellReference MonsterPosition,
                     ref CellReference PlayerPosition, ref CellReference
FlaskPosition, ref Boolean MonsterAwake, Boolean NewGame)
        {
            int Count;
            ResetCavern(Cavern);
            if (NewGame)
            {
                PlayerPosition.NoOfCellsSouth = 0;
                PlayerPosition.NoOfCellsEast = 0;
                MonsterAwake = false;
            }
            for (Count = 0; Count < NO_OF_TRAPS; Count++)
            {
                SetPositionOfItem(Cavern, ref TrapPositions[Count], 'T', NewGame);
            }
            SetPositionOfItem(Cavern, ref MonsterPosition, 'M', NewGame);
            SetPositionOfItem(Cavern, ref FlaskPosition, 'F', NewGame);
            SetPositionOfItem(Cavern, ref PlayerPosition, '*', NewGame);
        }

        public static void SetUpTrainingGame(char[,] Cavern, CellReference[]
TrapPositions, ref CellReference MonsterPosition, ref CellReference
PlayerPosition, ref CellReference FlaskPosition, ref Boolean MonsterAwake)
        {
            ResetCavern(Cavern);
            PlayerPosition.NoOfCellsSouth = 2;
```

```
            PlayerPosition.NoOfCellsEast = 4;
            MonsterAwake = false;
            TrapPositions[0].NoOfCellsSouth = 1;
            TrapPositions[0].NoOfCellsEast = 6;
            TrapPositions[1].NoOfCellsSouth = 3;
            TrapPositions[1].NoOfCellsEast = 4;
            MonsterPosition.NoOfCellsSouth = 0;
            MonsterPosition.NoOfCellsEast = 3;
            FlaskPosition.NoOfCellsSouth = 4;
            FlaskPosition.NoOfCellsEast = 5;
            SetUpGame(Cavern, TrapPositions, ref MonsterPosition, ref
PlayerPosition, ref FlaskPosition, ref MonsterAwake, false);
        }

        public static void LoadGame(CellReference[] TrapPositions, ref
CellReference MonsterPosition, ref CellReference PlayerPosition, ref CellReference
FlaskPosition, ref Boolean MonsterAwake)
        {
            string Filename;
            BinaryReader br;
            Console.Write("Enter the name of the file to load: ");
            Filename = Console.ReadLine();
            Console.WriteLine();
            br = new BinaryReader(new FileStream(Filename, FileMode.Open));
            TrapPositions[0].NoOfCellsSouth = br.ReadInt32();
            TrapPositions[0].NoOfCellsEast = br.ReadInt32();
            TrapPositions[1].NoOfCellsSouth = br.ReadInt32();
            TrapPositions[1].NoOfCellsEast = br.ReadInt32();
            MonsterPosition.NoOfCellsSouth = br.ReadInt32();
            MonsterPosition.NoOfCellsEast = br.ReadInt32();
            PlayerPosition.NoOfCellsSouth = br.ReadInt32();
            PlayerPosition.NoOfCellsEast = br.ReadInt32();
            FlaskPosition.NoOfCellsSouth = br.ReadInt32();
            FlaskPosition.NoOfCellsEast = br.ReadInt32();
            MonsterAwake = br.ReadBoolean();
            br.Close();
        }

        public static void SaveGame(CellReference[] TrapPositions, CellReference
MonsterPosition, CellReference PlayerPosition, CellReference FlaskPosition,
Boolean MonsterAwake)
        {
            string Filename;
            BinaryWriter bw;
            Console.Write("Enter new file name: ");
            Filename = Console.ReadLine();
            Console.WriteLine();
            bw = new BinaryWriter(new FileStream(Filename, FileMode.Create));
            bw.Write(TrapPositions[0].NoOfCellsSouth);
            bw.Write(TrapPositions[0].NoOfCellsEast);
            bw.Write(TrapPositions[1].NoOfCellsSouth);
            bw.Write(TrapPositions[1].NoOfCellsEast);
            bw.Write(MonsterPosition.NoOfCellsSouth);
            bw.Write(MonsterPosition.NoOfCellsEast);
```

```
            bw.Write(PlayerPosition.NoOfCellsSouth);
            bw.Write(PlayerPosition.NoOfCellsEast);
            bw.Write(FlaskPosition.NoOfCellsSouth);
            bw.Write(FlaskPosition.NoOfCellsEast);
            bw.Write(MonsterAwake);
            bw.Close();
        }

        public static void DisplayCavern(char[,] Cavern, Boolean MonsterAwake)
        {
            int Count1;
            int Count2;
            for (Count1 = 0; Count1 < N_S_DISTANCE; Count1++)
            {
                Console.WriteLine(" ------------- ");
                for (Count2 = 0; Count2 < W_E_DISTANCE; Count2++)
                {
                    if (Cavern[Count1, Count2] == ' ' || Cavern[Count1, Count2] ==
'*' || ((Cavern[Count1, Count2] == 'M') && MonsterAwake))
                    {
                        Console.Write("|" + Cavern[Count1, Count2]);
                    }
                    else
                    {
                        Console.Write("| ");
                    }
                }
                Console.WriteLine("|");
            }
            Console.WriteLine(" ------------- ");
            Console.WriteLine();
        }

        public static void DisplayMoveOptions()
        {
            Console.WriteLine();
            Console.WriteLine("Enter N to move NORTH");
            Console.WriteLine("Enter E to move EAST");
            Console.WriteLine("Enter S to move SOUTH");
            Console.WriteLine("Enter W to move WEST");
            Console.WriteLine("Enter M to return to the Main Menu");
            Console.WriteLine();
        }

        public static char GetMove()
        {
            char Move;
            Move = char.Parse(Console.ReadLine());
            Console.WriteLine();
            return Move;
        }
```

```
        public static void MakeMove(char[,] Cavern, char Direction, ref
CellReference PlayerPosition)
        {
            Cavern[PlayerPosition.NoOfCellsSouth, PlayerPosition.NoOfCellsEast] =
' ';
            switch (Direction)
            {
                case 'N':
                    PlayerPosition.NoOfCellsSouth = PlayerPosition.NoOfCellsSouth
- 1;
                    break;
                case 'S':
                    PlayerPosition.NoOfCellsSouth = PlayerPosition.NoOfCellsSouth
+ 1;
                    break;
                case 'W':
                    PlayerPosition.NoOfCellsEast = PlayerPosition.NoOfCellsEast -
1;
                    break;
                case 'E':
                    PlayerPosition.NoOfCellsEast = PlayerPosition.NoOfCellsEast +
1;
                    break;
            }
            Cavern[PlayerPosition.NoOfCellsSouth, PlayerPosition.NoOfCellsEast] =
'*';
        }

        public static Boolean CheckValidMove(CellReference PlayerPosition, char
Direction)
        {
            Boolean ValidMove;
            ValidMove = true;
            if (!(Direction == 'N' || Direction == 'S' || Direction == 'W' ||
Direction == 'E' || Direction == 'M'))
            {
                ValidMove = false;
            }
            return ValidMove;
        }

        public static Boolean CheckIfSameCell(CellReference FirstCellPosition,
CellReference SecondCellPosition)
        {
            Boolean InSameCell = false;
            if (FirstCellPosition.NoOfCellsSouth ==
SecondCellPosition.NoOfCellsSouth && FirstCellPosition.NoOfCellsEast ==
SecondCellPosition.NoOfCellsEast)
            {
                InSameCell = true;
            }
            return InSameCell;
        }
```

```
        public static void DisplayWonGameMessage()
        {
            Console.WriteLine("Well Done!  You have found the flask containing the
Styxian potion.");
            Console.WriteLine("You have won the game of MONSTER!");
            Console.WriteLine();
        }

        public static void DisplayTrapMessage()
        {
            Console.WriteLine("Oh no!  You have set off a trap.  Watch out, the
monster is now awake!");
            Console.WriteLine();
        }

        public static void MoveFlask(char[,] Cavern, CellReference
NewCellForFlask, ref CellReference FlaskPosition)
        {
            Cavern[NewCellForFlask.NoOfCellsSouth, NewCellForFlask.NoOfCellsEast]
= 'F';
            Cavern[FlaskPosition.NoOfCellsSouth, FlaskPosition.NoOfCellsEast] = '
';
            FlaskPosition = NewCellForFlask;
        }

        public static void MakeMonsterMove(char[,] Cavern, ref CellReference
MonsterPosition, ref CellReference FlaskPosition, CellReference PlayerPosition)
        {
            CellReference OriginalMonsterPosition = new CellReference();
            Boolean MonsterMovedToSameCellAsFlask = false;
            OriginalMonsterPosition.NoOfCellsSouth =
MonsterPosition.NoOfCellsSouth;
            OriginalMonsterPosition.NoOfCellsEast = MonsterPosition.NoOfCellsEast;
            Cavern[MonsterPosition.NoOfCellsSouth, MonsterPosition.NoOfCellsEast]
= ' ';
            if (MonsterPosition.NoOfCellsSouth < PlayerPosition.NoOfCellsSouth)
            {
                MonsterPosition.NoOfCellsSouth = MonsterPosition.NoOfCellsSouth +
1;
            }
            else if (MonsterPosition.NoOfCellsSouth >
PlayerPosition.NoOfCellsSouth)
            {
                MonsterPosition.NoOfCellsSouth = MonsterPosition.NoOfCellsSouth -
1;
            }
            else if (MonsterPosition.NoOfCellsEast < PlayerPosition.NoOfCellsEast)
            {
                MonsterPosition.NoOfCellsEast = MonsterPosition.NoOfCellsEast + 1;
            }
            else if (MonsterPosition.NoOfCellsEast > PlayerPosition.NoOfCellsEast)
            {
                MonsterPosition.NoOfCellsEast = MonsterPosition.NoOfCellsEast - 1;
```

```
            }
            MonsterMovedToSameCellAsFlask = CheckIfSameCell(MonsterPosition,
FlaskPosition);
            if (MonsterMovedToSameCellAsFlask)
            {
                MoveFlask(Cavern, OriginalMonsterPosition, ref FlaskPosition);
            }
            Cavern[MonsterPosition.NoOfCellsSouth, MonsterPosition.NoOfCellsEast]
= 'M';
        }

        public static void DisplayLostGameMessage()
        {
            Console.WriteLine("ARGHHHHHH!  The monster has eaten you.  GAME
OVER.");
            Console.WriteLine("Maybe you will have better luck the next time you
play MONSTER!");
            Console.WriteLine();
        }

        public static void PlayGame(char[,] Cavern, CellReference[] TrapPositions,
ref CellReference MonsterPosition, ref CellReference PlayerPosition, ref
CellReference FlaskPosition, ref Boolean MonsterAwake)
        {
            Boolean Eaten = false;
            Boolean FlaskFound = false;
            Boolean ValidMove = false;
            int Count = 0;
            char MoveDirection = ' ';
            DisplayCavern(Cavern, MonsterAwake);
            while (!(Eaten || FlaskFound || MoveDirection == 'M'))
            {
                ValidMove = false;
                while (!ValidMove)
                {
                    DisplayMoveOptions();
                    MoveDirection = GetMove();
                    ValidMove = CheckValidMove(PlayerPosition, MoveDirection);
                }
                if (MoveDirection != 'M')
                {
                    MakeMove(Cavern, MoveDirection, ref PlayerPosition);
                    DisplayCavern(Cavern, MonsterAwake);
                    FlaskFound = CheckIfSameCell(PlayerPosition, FlaskPosition);
                    if (FlaskFound)
                    {
                        DisplayWonGameMessage();
                    }
                    Eaten = CheckIfSameCell(MonsterPosition, PlayerPosition);
                    if (!MonsterAwake && !FlaskFound && !Eaten)
                    {
                        MonsterAwake = CheckIfSameCell(PlayerPosition,
TrapPositions[0]);
                        if (!MonsterAwake)
```

```
                        {
                            MonsterAwake = CheckIfSameCell(PlayerPosition,
TrapPositions[1]);
                        }
                        if (MonsterAwake)
                        {
                            DisplayTrapMessage();
                            DisplayCavern(Cavern, MonsterAwake);
                        }
                    }
                    if (MonsterAwake && !Eaten && !FlaskFound)
                    {
                        Count = 0;
                        while (Count < 2 && !Eaten)
                        {
                            MakeMonsterMove(Cavern, ref MonsterPosition, ref
FlaskPosition, PlayerPosition);
                            Eaten = CheckIfSameCell(MonsterPosition,
PlayerPosition);
                            Console.WriteLine();
                            Console.WriteLine("Press Enter key to continue");
                            Console.ReadLine();
                            DisplayCavern(Cavern, MonsterAwake);
                            Count = Count + 1;
                        }
                    }
                    if (Eaten)
                    {
                        DisplayLostGameMessage();
                    }
                }
            }
        }

    }
}
```