



SPECIMEN MATERIAL

A-level
COMPUTER SCIENCE
(7517/1D)

Paper 1 Python

Skeleton Program

Python 2.7

```
import random

class CellReference:
    def __init__(self):
        self.NoOfCellsEast = 0
        self.NoOfCellsSouth = 0

class Item:
    def __init__(self):
        self.NoOfCellsEast = 0
        self.NoOfCellsSouth = 0

    def CheckIfSameCell(self, Position):
        if (self.NoOfCellsEast == Position.NoOfCellsEast) and \
            (self.NoOfCellsSouth == Position.NoOfCellsSouth):
            return True
        else:
            return False

    def GetPosition(self):
        Position = CellReference()
        Position.NoOfCellsEast = self.NoOfCellsEast
        Position.NoOfCellsSouth = self.NoOfCellsSouth
        return Position

    def SetPosition(self, Position):
        self.NoOfCellsEast = Position.NoOfCellsEast
        self.NoOfCellsSouth = Position.NoOfCellsSouth

class Character(Item):
    def __init__(self):
        Item.__init__(self)

    def MakeMove(self, Direction):
        if Direction == 'N':
            self.NoOfCellsSouth = self.NoOfCellsSouth - 1
        elif Direction == 'S':
            self.NoOfCellsSouth = self.NoOfCellsSouth + 1
        elif Direction == 'W':
            self.NoOfCellsEast = self.NoOfCellsEast - 1
        elif Direction == 'E':
            self.NoOfCellsEast = self.NoOfCellsEast + 1

class Enemy(Item):
    def __init__(self):
        Item.__init__(self)
        self.Awake = False

    def ChangeSleepStatus(self):
        self.Awake = not self.Awake
```

```
def GetAwake(self):
    return self.Awake

def MakeMove(self, PlayerPosition):
    if self.NoOfCellsSouth < PlayerPosition.NoOfCellsSouth:
        self.NoOfCellsSouth = self.NoOfCellsSouth + 1
    elif self.NoOfCellsSouth > PlayerPosition.NoOfCellsSouth:
        self.NoOfCellsSouth = self.NoOfCellsSouth - 1
    elif self.NoOfCellsEast < PlayerPosition.NoOfCellsEast:
        self.NoOfCellsEast = self.NoOfCellsEast + 1
    else:
        self.NoOfCellsEast = self.NoOfCellsEast - 1

class Trap(Item):
    def __init__(self):
        Item.__init__(self)
        self.Triggered = False

    def GetTriggered(self):
        return self.Triggered

    def ToggleTrap(self):
        self.Triggered = not self.Triggered

class Grid:
    def __init__(self, NS, WE):
        self.NSDistance = NS
        self.WEDistance = WE
        self.CavernState = []
        for Count1 in range(self.NSDistance + 1):
            BlankRow = []
            for Count2 in range(self.WEDistance + 1):
                BlankRow.append(' ')
            self.CavernState.append(BlankRow)

    def Display(self, MonsterAwake):
        print " ----- "
        for Count1 in range(0, self.NSDistance + 1):
            Row = ""
            for Count2 in range(0, self.WEDistance + 1):
                if (self.CavernState[Count1][Count2] == ' ') or \
                    (self.CavernState[Count1][Count2] == '*') or \
                    ((self.CavernState[Count1][Count2] == 'M') and MonsterAwake):
                    Row = Row + '|' + self.CavernState[Count1][Count2]
                else:
                    Row = Row + '| '
            print Row + '|'
            print " ----- "
        print

    def PlaceItem(self, Position, Item):
        self.CavernState[Position.NoOfCellsSouth][Position.NoOfCellsEast] = Item

    def IsCellEmpty(self, Position):
```

```
if self.CavernState[Position.NoOfCellsSouth][Position.NoOfCellsEast] == ' ':
    return True
else:
    return False

def Reset(self):
    for Count1 in range(self.NSDistance + 1):
        for Count2 in range(self.WEDistance + 1):
            self.CavernState[Count1][Count2] = ' '

class Game:
    NS = 4
    WE = 6

    def __init__(self, IsATrainingGame):
        self.Player = Character()
        self.Cavern = Grid(Game.NS, Game.WE)
        self.Monster = Enemy()
        self.Flask = Item()
        self.Trap1 = Trap()
        self.Trap2 = Trap()
        self.TrainingGame = IsATrainingGame
        random.seed()
        self.SetUpGame()
        self.Play()

    def Play(self):
        Eaten = False
        FlaskFound = False
        MoveDirection = ' '
        Position = CellReference()
        self.Cavern.Display(self.Monster.GetAwake())
        while not Eaten and not FlaskFound and (MoveDirection != 'M'):
            ValidMove = False
            while not ValidMove:
                self.DisplayMoveOptions()
                MoveDirection = self.GetMove()
                ValidMove = self.CheckValidMove(MoveDirection)
            if MoveDirection != 'M':
                self.Cavern.PlaceItem(self.Player.GetPosition(), ' ')
                self.Player.MakeMove(MoveDirection)
                self.Cavern.PlaceItem(self.Player.GetPosition(), '*')
                self.Cavern.Display(self.Monster.GetAwake())
                FlaskFound = self.Player.CheckIfSameCell(self.Flask.GetPosition())
                if FlaskFound:
                    self.DisplayWonGameMessage()
                Eaten = self.Monster.CheckIfSameCell(self.Player.GetPosition())
                # This selection structure checks to see if the player has triggered one
of the traps in the cavern
                if not self.Monster.GetAwake() and \
                    not FlaskFound and not Eaten and \
                    (self.Player.CheckIfSameCell(self.Trap1.GetPosition()) and \
                     not self.Trap1.GetTriggered() or \
```

```
        self.Player.CheckIfSameCell(self.Trap2.GetPosition()) and \
        not self.Trap2.GetTriggered()):
    self.Monster.ChangeSleepStatus()
    self.DisplayTrapMessage()
    self.Cavern.Display(self.Monster.GetAwake())
if (self.Monster.GetAwake()) and not Eaten and not FlaskFound:
    Count = 0
    while Count != 2 and not Eaten:
        self.Cavern.PlaceItem(self.Monster.GetPosition(), ' ')
        Position = self.Monster.GetPosition()
        self.Monster.MakeMove(self.Player.GetPosition())
        self.Cavern.PlaceItem(self.Monster.GetPosition(), 'M')
        if self.Monster.CheckIfSameCell(self.Flask.GetPosition()):
            self.Flask.SetPosition(Position)
            self.Cavern.PlaceItem(Position, 'F')
        Eaten = self.Monster.CheckIfSameCell(self.Player.GetPosition())
        print
        print "Press Enter key to continue"
        raw_input()
        self.Cavern.Display(self.Monster.GetAwake())
        Count = Count + 1
    if Eaten:
        self.DisplayLostGameMessage()

def DisplayMoveOptions(self):
    print
    print "Enter N to move NORTH"
    print "Enter S to move SOUTH"
    print "Enter E to move EAST"
    print "Enter W to move WEST"
    print "Enter M to return to the Main Menu"
    print

def GetMove(self):
    Move = raw_input()
    print
    if Move != "":
        return Move[0]
    else:
        return ""

def DisplayWonGameMessage(self):
    print "Well done! You have found the flask containing the Styxian potion."
    print "You have won the game of MONSTER!"
    print

def DisplayTrapMessage(self):
    print "Oh no! You have set off a trap. Watch out, the monster is now awake!"
    print

def DisplayLostGameMessage(self):
    print "ARGHHHHHHH! The monster has eaten you. GAME OVER."
    print "Maybe you will have better luck next time you play MONSTER!"
    print
```

```
def CheckValidMove(self, Direction):
    ValidMove = True
    if not(Direction in ['N', 'S', 'W', 'E', 'M']):
        ValidMove = False
    return ValidMove

def SetPositionOfItem(self, Item):
    Position = self.GetNewRandomPosition()
    while not self.Cavern.IsCellEmpty(Position):
        Position = self.GetNewRandomPosition()
    self.Cavern.PlaceItem(Position, Item)
    return Position

def SetUpGame(self):
    Position = CellReference()
    self.Cavern.Reset()
    if not self.TrainingGame:
        Position.NoOfCellsEast = 0
        Position.NoOfCellsSouth = 0
        self.Player.SetPosition(Position)
        self.Cavern.PlaceItem(Position, '*')
        self.Trap1.SetPosition(self.SetPositionOfItem('T'))
        self.Trap2.SetPosition(self.SetPositionOfItem('T'))
        self.Monster.SetPosition(self.SetPositionOfItem('M'))
        self.Flask.SetPosition(self.SetPositionOfItem('F'))
    else:
        Position.NoOfCellsEast = 4
        Position.NoOfCellsSouth = 2
        self.Player.SetPosition(Position)
        self.Cavern.PlaceItem(Position, '*')
        Position.NoOfCellsEast = 6
        Position.NoOfCellsSouth = 2
        self.Trap1.SetPosition(Position)
        self.Cavern.PlaceItem(Position, 'T')
        Position.NoOfCellsEast = 4
        Position.NoOfCellsSouth = 3
        self.Trap2.SetPosition(Position)
        self.Cavern.PlaceItem(Position, 'T')
        Position.NoOfCellsEast = 4
        Position.NoOfCellsSouth = 0
        self.Monster.SetPosition(Position)
        self.Cavern.PlaceItem(Position, 'M')
        Position.NoOfCellsEast = 3
        Position.NoOfCellsSouth = 1
        self.Flask.SetPosition(Position)
        self.Cavern.PlaceItem(Position, 'F')

def GetNewRandomPosition(self):
    Position = CellReference()
    while (Position.NoOfCellsSouth == 0) and (Position.NoOfCellsEast == 0):
        Position.NoOfCellsSouth = random.randint(0, Game.NS)
        Position.NoOfCellsEast = random.randint(0, Game.WE)
```

```
    return Position

def DisplayMenu():
    print "MAIN MENU"
    print
    print "1. Start new game"
    print "2. Play training game"
    print "9. Quit"
    print
    print "Please enter your choice:",

def GetMainMenuChoice():
    Choice = int(raw_input())
    print
    return Choice

if __name__ == "__main__":
    Choice = 0
    while Choice != 9:
        DisplayMenu()
        Choice = GetMainMenuChoice()
        if Choice == 1:
            MyGame = Game(False)
        elif Choice == 2:
            MyGame = Game(True)
```

Python 3.3

```
import random

class CellReference:
    def __init__(self):
        self.NoOfCellsEast = 0
        self.NoOfCellsSouth = 0

class Item:
    def __init__(self):
        self.NoOfCellsEast = 0
        self.NoOfCellsSouth = 0

    def CheckIfSameCell(self, Position):
        if (self.NoOfCellsEast == Position.NoOfCellsEast) and \
            (self.NoOfCellsSouth == Position.NoOfCellsSouth):
            return True
        else:
            return False

    def GetPosition(self):
        Position = CellReference()
        Position.NoOfCellsEast = self.NoOfCellsEast
        Position.NoOfCellsSouth = self.NoOfCellsSouth
        return Position

    def SetPosition(self, Position):
        self.NoOfCellsEast = Position.NoOfCellsEast
        self.NoOfCellsSouth = Position.NoOfCellsSouth

class Character(Item):
    def __init__(self):
        Item.__init__(self)

    def MakeMove(self, Direction):
        if Direction == 'N':
            self.NoOfCellsSouth = self.NoOfCellsSouth - 1
        elif Direction == 'S':
            self.NoOfCellsSouth = self.NoOfCellsSouth + 1
        elif Direction == 'W':
            self.NoOfCellsEast = self.NoOfCellsEast - 1
        elif Direction == 'E':
            self.NoOfCellsEast = self.NoOfCellsEast + 1

class Enemy(Item):
    def __init__(self):
        Item.__init__(self)
        self.Awake = False

    def ChangeSleepStatus(self):
        self.Awake = not self.Awake

    def GetAwake(self):
```

```
    return self.Awake

def MakeMove(self, PlayerPosition):
    if self.NoOfCellsSouth < PlayerPosition.NoOfCellsSouth:
        self.NoOfCellsSouth = self.NoOfCellsSouth + 1
    elif self.NoOfCellsSouth > PlayerPosition.NoOfCellsSouth:
        self.NoOfCellsSouth = self.NoOfCellsSouth - 1
    elif self.NoOfCellsEast < PlayerPosition.NoOfCellsEast:
        self.NoOfCellsEast = self.NoOfCellsEast + 1
    else:
        self.NoOfCellsEast = self.NoOfCellsEast - 1

class Trap(Item):
    def __init__(self):
        Item.__init__(self)
        self.Triggered = False

    def GetTriggered(self):
        return self.Triggered

    def ToggleTrap(self):
        self.Triggered = not self.Triggered

class Grid:
    def __init__(self, NS, WE):
        self.NSDistance = NS
        self.WEDistance = WE
        self.CavernState = []
        for Count1 in range(self.NSDistance + 1):
            BlankRow = []
            for Count2 in range(self.WEDistance + 1):
                BlankRow.append(' ')
            self.CavernState.append(BlankRow)

    def Display(self, MonsterAwake):
        print(" ----- ")
        for Count1 in range(self.NSDistance + 1):
            for Count2 in range(self.WEDistance + 1):
                if (self.CavernState[Count1][Count2] == ' ') or \
                    (self.CavernState[Count1][Count2] == '*') or \
                    ((self.CavernState[Count1][Count2] == 'M') and MonsterAwake):
                    print('|' + self.CavernState[Count1][Count2], end='')
                else:
                    print('| ', end='')
            print('|')
            print(" ----- ")
        print()

    def PlaceItem(self, Position, Item):
        self.CavernState[Position.NoOfCellsSouth][Position.NoOfCellsEast] = Item

    def IsCellEmpty(self, Position):
        if Position.NoOfCellsEast < 0 or Position.NoOfCellsEast > Game.WE:
            print("Error")
```

```
if Position.NoOfCellsSouth < 0 or Position.NoOfCellsSouth > Game.NS:
    print("Error")
if self.CavernState[Position.NoOfCellsSouth][Position.NoOfCellsEast] == ' ':
    return True
else:
    return False

def Reset(self):
    for Count1 in range(self.NSDistance + 1):
        for Count2 in range(self.WEDistance + 1):
            self.CavernState[Count1][Count2] = ' '

class Game:
    NS = 4
    WE = 6

    def __init__(self, IsATrainingGame):
        self.Player = Character()
        self.Cavern = Grid(Game.NS, Game.WE)
        self.Monster = Enemy()
        self.Flask = Item()
        self.Trap1 = Trap()
        self.Trap2 = Trap()
        self.TrainingGame = IsATrainingGame
        random.seed()
        self.SetUpGame()
        self.Play()

    def Play(self):
        Eaten = False
        FlaskFound = False
        MoveDirection = ' '
        Position = CellReference()
        self.Cavern.Display(self.Monster.GetAwake())
        while not Eaten and not FlaskFound and (MoveDirection != 'M'):
            ValidMove = False
            while not ValidMove:
                self.DisplayMoveOptions()
                MoveDirection = self.GetMove()
                ValidMove = self.CheckValidMove(MoveDirection)
            if MoveDirection != 'M':
                self.Cavern.PlaceItem(self.Player.GetPosition(), ' ')
                self.Player.MakeMove(MoveDirection)
                self.Cavern.PlaceItem(self.Player.GetPosition(), '*')
                self.Cavern.Display(self.Monster.GetAwake())
                FlaskFound = self.Player.CheckIfSameCell(self.Flask.GetPosition())
                if FlaskFound:
                    self.DisplayWonGameMessage()
                Eaten = self.Monster.CheckIfSameCell(self.Player.GetPosition())
                # This selection structure checks to see if the player has triggered one
                # of the traps in the cavern
                if not self.Monster.GetAwake() and \
                    not FlaskFound and not Eaten and \
```

```
(self.Player.CheckIfSameCell(self.Trap1.GetPosition()) and \
not self.Trap1.GetTriggered() or \
self.Player.CheckIfSameCell(self.Trap2.GetPosition()) and \
not self.Trap2.GetTriggered()):
self.Monster.ChangeSleepStatus()
self.DisplayTrapMessage()
self.Cavern.Display(self.Monster.GetAwake())
if (self.Monster.GetAwake()) and not Eaten and not FlaskFound:
    Count = 0
    while Count != 2 and not Eaten:
        self.Cavern.PlaceItem(self.Monster.GetPosition(), ' ')
        Position = self.Monster.GetPosition()
        self.Monster.MakeMove(self.Player.GetPosition())
        self.Cavern.PlaceItem(self.Monster.GetPosition(), 'M')
        if self.Monster.CheckIfSameCell(self.Flask.GetPosition()):
            self.Flask.SetPosition(Position)
            self.Cavern.PlaceItem(Position, 'F')
        Eaten = self.Monster.CheckIfSameCell(self.Player.GetPosition())
        print()
        print('Press Enter key to continue')
        input()
        self.Cavern.Display(self.Monster.GetAwake())
        Count = Count + 1
    if Eaten:
        self.DisplayLostGameMessage()

def DisplayMoveOptions(self):
    print()
    print("Enter N to move NORTH")
    print("Enter S to move SOUTH")
    print("Enter E to move EAST")
    print("Enter W to move WEST")
    print("Enter M to return to the Main Menu")
    print()

def GetMove(self):
    Move = input()
    print()
    if Move != "":
        return Move[0]
    else:
        return ""

def DisplayWonGameMessage(self):
    print("Well done! You have found the flask containing the Styxian potion.")
    print("You have won the game of MONSTER!")
    print()

def DisplayTrapMessage(self):
    print("Oh no! You have set off a trap. Watch out, the monster is now
awake!")
    print()

def DisplayLostGameMessage(self):
```

```
print("ARGHHHHHHH! The monster has eaten you. GAME OVER.")
print("Maybe you will have better luck next time you play MONSTER!")
print()

def CheckValidMove(self, Direction):
    ValidMove = True
    if not(Direction in ['N', 'S', 'W', 'E', 'M']):
        ValidMove = False
    return ValidMove

def SetPositionOfItem(self, Item):
    Position = self.GetNewRandomPosition()
    while not self.Cavern.IsCellEmpty(Position):
        Position = self.GetNewRandomPosition()
    self.Cavern.PlaceItem(Position, Item)
    return Position

def SetUpGame(self):
    Position = CellReference()
    self.Cavern.Reset()
    if not self.TrainingGame:
        Position.NoOfCellsEast = 0
        Position.NoOfCellsSouth = 0
        self.Player.SetPosition(Position)
        self.Cavern.PlaceItem(Position, '*')
        self.Trap1.SetPosition(self.SetPositionOfItem('T'))
        self.Trap2.SetPosition(self.SetPositionOfItem('T'))
        self.Monster.SetPosition(self.SetPositionOfItem('M'))
        self.Flask.SetPosition(self.SetPositionOfItem('F'))
    else:
        Position.NoOfCellsEast = 4
        Position.NoOfCellsSouth = 2
        self.Player.SetPosition(Position)
        self.Cavern.PlaceItem(Position, '*')
        Position.NoOfCellsEast = 6
        Position.NoOfCellsSouth = 2
        self.Trap1.SetPosition(Position)
        self.Cavern.PlaceItem(Position, 'T')
        Position.NoOfCellsEast = 4
        Position.NoOfCellsSouth = 3
        self.Trap2.SetPosition(Position)
        self.Cavern.PlaceItem(Position, 'T')
        Position.NoOfCellsEast = 4
        Position.NoOfCellsSouth = 0
        self.Monster.SetPosition(Position)
        self.Cavern.PlaceItem(Position, 'M')
        Position.NoOfCellsEast = 3
        Position.NoOfCellsSouth = 1
        self.Flask.SetPosition(Position)
        self.Cavern.PlaceItem(Position, 'F')

def GetNewRandomPosition(self):
    Position = CellReference()
```

```
while (Position.NoOfCellsSouth == 0) and (Position.NoOfCellsEast == 0):
    Position.NoOfCellsSouth = random.randint(0, Game.NS)
    Position.NoOfCellsEast = random.randint(0, Game.WE)
return Position

def DisplayMenu():
    print("MAIN MENU")
    print()
    print("1. Start new game")
    print("2. Play training game")
    print("9. Quit")
    print()
    print("Please enter your choice: ", end='')

def GetMainMenuChoice():
    Choice = int(input())
    print()
    return Choice

if __name__ == "__main__":
    Choice = 0
    while Choice != 9:
        DisplayMenu()
        Choice = GetMainMenuChoice()
        if Choice == 1:
            MyGame = Game(False)
        elif Choice == 2:
            MyGame = Game(True)
```

