# Monster Questions

(a)     List the Identifiers for the Classes in the Monster Skeleton Program

**(5)**

 Game, Grid, Enemy, Character, Trap, Item

The Python version also has class CellReference. This is implemented as a Structure in VB and C#.

(b)     Give an example of instantiation from the **Skeleton Program**.

**(1)**

Private Player As New Character ‘ VB

 self.\_\_Player = Character() // Python

 private Character Player = new Character(); // C#

 (c)     List all the methods from the Game class that will return one or more values

**(4)**

GetMove, CheckValidMove, SetPositionOfItem, GetNewRandomPosition

(d)     State the name of an identifier for a class that uses composition.

**(1)**

 **Game**

(e)     State the name of an identifier for a local variable that is used to store a Boolean value.

**(1)**

 Eaten (in Game.Play method)

(f)     List all the identifiers of the objects that are examples of compositional aggregation from the Skeleton Program.
Class: Game

Player (Character), Cavern (Grid), Monster (Enemy), Flask (Item), Trap1 (Trap) and Trap2 (Trap).

**(1)**

 (g)     Look at the Play subroutine in the Game class in the **Skeleton Program**.

Explain the conditions that must be met for the loop that ends on the last line of this method to be exited

The loop will exit if the player is Eaten (by the monster - i.e. the player loses), the Flask is found (i.e. player wins) or the player has pressed M to exit the game and return to the main menu

**(1)**

 (h)   Lists the identifiers for all the Constants in the **Skeleton Program.** Give the benefits of using constants over numerical values and variable initialised with a value.

NS and WE

Benefits are:
Constants can be used throughout the code so you don’t need to remember the value (i.e. the size of the grid).The values of the constants can be changed by the programmer and the new size of the grid will be reflected in the rest of the code without any other changes needed to be made.
The programmer cannot accidentally change the value
The code is more readable

**(4)**

(i)     List the all the example of inheritance State the base and subclasses identifiers in each case

Base Class: Item, Sub Classes: Trap, Enemy, Character

 **(5)**

(j) List all the examples of polymorphism. State the class and method identifiers involved.

Visual Basic and C# have the Overridable qualifier:

Class: MakeMove: Methods: MakeMove and ChangeSleepStatus

In Python all methods are overridable so the above answer is also true

 **(4)**

(k) In the code give the identifier of a property or method that is:
Protected: NoOfCellsEast
Private: Trap1
Public (Play) **(4)**

(l) Explain the difference in a property that have been declared Private or Protected

The Private access modifier limits the scope of the variable, attribute or method to the class within which is is declared, whereas the Protected access modifer limits the scope to the class in which it is declared, and al its subclasses.

(3)

(m) Describe the changes that would need to be made to the code to change the game so that when a player reached the east-most square they could continue to move east and would appear on the west-most square (like in pac man).

In the MakeMove method there should be a selection at the end to determine if the player has gone beyond the top, bottom, left or right edge of the grid. If they have then the position should be set to the opposite edge as follows:

If NoOfCellsEast = -1 then NoOfCellsEast = EW
If NoOfCellsEast = EW+1 then NoOfCellsEast = 0
If NoOfCellsSouth = -1 then NoOfCellsSouth = NS
If NoOfCellsSouth = NS+1 then NoOfCellsSouth = 0
 (4)

(n) Describe the changes to the code required to have a magic potion that if the player gain would allow them to make 2 moves per turn.

This is best done in the Loop Until Valid loop (While Not ValidMove in Python) in the Play method.

Add an attribute that is set if the Player picks up the magic potion and add a counter to this loop, so the continue condition would be:
While Not ValidMove And (Not Player.HasMagicPotion Or Player.HasMagicPotion and NoOfMoves<Count)

 (6)