Guru Nanak College - Budhlada

Semester – l
[2023 – 2024]

Project Management Assignment File

Submitted To:	*Submitted By:*
Assistant Professor	Name: **Mitesh Kumar**
Manwinder Singh	Class: **B.Voc(SD) – 1**
	Roll No.: **11116**

INDEX

DECLARATION

I hereby declare that the Project Report entitled "Snake Game" in MS Excel is an authentic record of my own work as requirements for the degree of B.Voc. (Software Development), Guru Nanak College, Budhlada, under the guidance of Asst. Prof. Manwinder sir.

Date: _____

Signature: _____

B.Voc. (Software Development)-1

Roll no.: _____

CERTIFICATE

This is to certify that Mr. / Ms. _____ has partially completed Project as a Partial Fulfillment of Degree of Bachelor of Vocation in Software Development. He / She has completed project entitled "Snake Game in MS Excel".

_____

Signature

ACKNOWLEDGEMENTS

I would like to thank everyone who has contributed to the successful completion of this project. I would like to express my gratitude to my project supervisor, Asst. Prof. Manwinder sir for his invaluable advice, guidance and his enormous patience throughout the development of the project.

In addition, I would also like to express my gratitude to my loving parents and friends who have helped and given me encouragement.

Signature: _____

B.Voc. (Software Development)-1

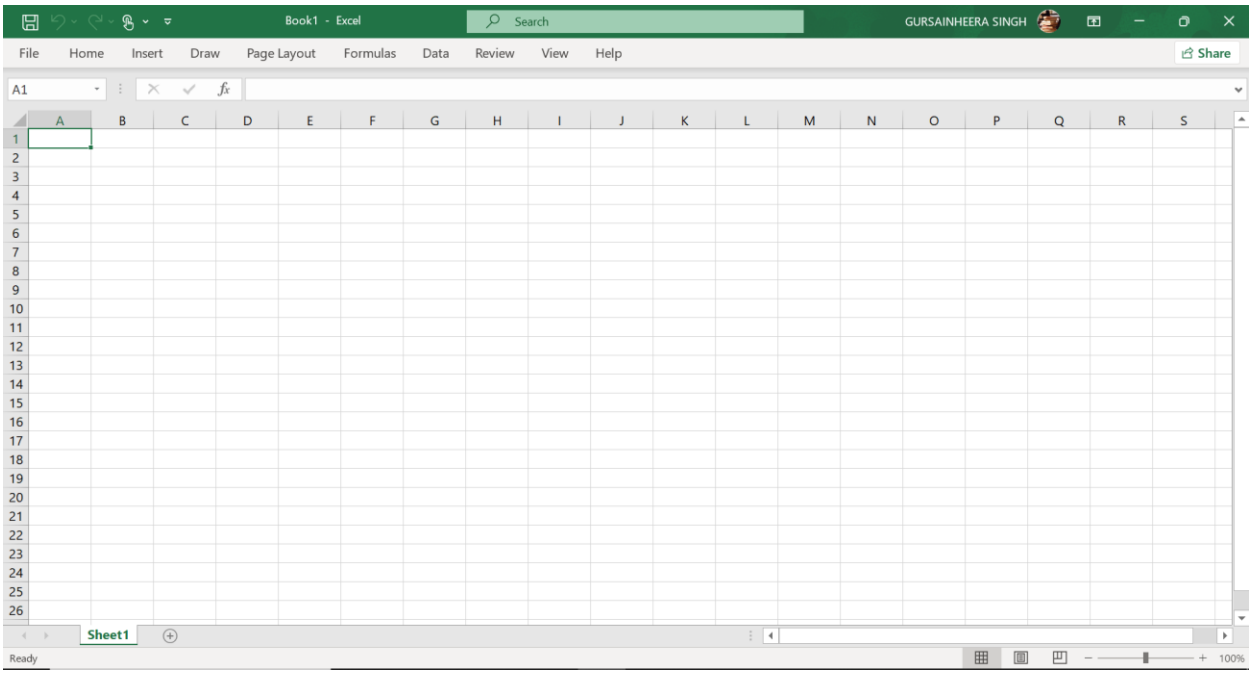
Roll no.: _____

INTRODUCTION

What does Microsoft Excel mean?

Microsoft Excel is a software program produced by Microsoft that allows users to organize, format and calculate data with formulas using a spreadsheet system. This software is part of the Microsoft Office suite and is compatible with other applications in the Office suite.

Excel is a commercial spreadsheet application produced and distributed by Microsoft for Microsoft Windows and Mac OS. It features the ability to perform basic calculations, use graphing tools, create pivot tables and create macros.



Excel has the same basic features as all spreadsheet applications, which use a collection of cells arranged into rows and columns to organize and manipulate data. They can also display data as charts, histograms and line graphs.

Excel permits users to arrange data so as to view various factors from different perspectives. Visual Basic is used for applications in Excel, allowing users to create a variety of complex numerical methods. Programmers are given an option to code directly using the Visual Basic Editor, including Windows for writing code, debugging and code module organization.

Features of MS Excel

Microsoft Excel is powerful spreadsheet software that offers a wide range of features to help users manage, analyze, and visualize data. Some of its main features include:

1. Grid Interface: Excel's primary interface is a grid of cells arranged in rows and columns. This grid allows you to input, organize, and manipulate data easily.

2. Data Entry and Formatting: Excel provides tools for entering and formatting data. You can customize fonts, styles, colors, and cell borders to make your data visually appealing and easy to read.

3. Formulas and Functions: Excel allows users to perform calculations on data using a vast library of built-in functions and mathematical operators. Functions like SUM, AVERAGE, IF, and VLOOKUP are widely used for data analysis.

4. Charts and Graphs: Excel enables users to create various types of charts and graphs to visualize data trends and patterns. Common chart types include bar charts, pie charts, line charts, and scatter plots.

5. Data Sorting and Filtering: You can easily sort and filter data in Excel to organize it in a meaningful way. This is helpful for finding specific information in large datasets.

6. Data Validation: Excel lets you set rules and validation criteria for data entry, ensuring that data is accurate and consistent.

7. Data Import and Export: You can import data from external sources like databases, text files, and the web. Excel also supports exporting data in various formats.

8. PivotTables: PivotTables are powerful tools for summarizing and analyzing large datasets. They allow you to create custom reports and perform in-depth data analysis.

9. Conditional Formatting: This feature allows you to apply formatting rules based on cell values. For example, you can highlight cells with values that meet specific conditions.

10. Data Protection: Excel offers password protection and encryption to secure your spreadsheets. You can restrict access to specific cells or sheets.

11. What-If Analysis: You can perform what-if analysis by changing input values and observing how they affect calculated results, which is useful for scenario planning.

12. Macros and Automation: Excel supports VBA (Visual Basic for Applications), allowing you to automate tasks and create custom functions and macros.

13. Collaboration: Excel enables real-time collaboration through cloud-based services like Microsoft 365, where multiple users can work on a spreadsheet simultaneously.

14. Data Validation: You can set rules to validate data entry, ensuring accuracy and consistency.

15. Solver: Excel includes a Solver tool for optimization and solving complex mathematical models.

16. Add-Ins: You can extend Excel's functionality by adding various add-ins and customizing the software to suit your needs.

17. Data Analysis Tools: Excel provides advanced data analysis tools, including regression analysis, data tables, and Goal Seek for modeling and decision-making.

These are some of the key features that make Microsoft Excel a versatile tool for data management, analysis, and reporting, making it a staple in businesses, educational institutions, and various industries.

Formulas

A formula is an expression which gives some result using one or more cells. Functions are predefined formulas and are already available in Excel. Microsoft Excel offers a wide range of basic formulas for performing calculations and data analysis. Here are some of the most commonly used basic formulas with examples:

1. SUM: Adds up a range of numbers.

Example: `=SUM(A1:A5)` adds the numbers in cells A1 to A5.

2. AVERAGE: Calculates the average of a range of numbers.

Example: `=AVERAGE(B1:B4)` calculates the average of the numbers in cells B1 to B4.

3. MAX: Returns the maximum value in a range.

Example: `=MAX(C1:C6)` finds the largest value in cells C1 to C6.

4. MIN: Returns the minimum value in a range.

Example: `=MIN(D1:D3)` finds the smallest value in cells D1 to D3.

5. COUNT: Counts the number of cells in a range that contain numbers.

Example: `=COUNT(E1:E7)` counts how many cells in E1 to E7 contain numbers.

6. IF: Performs a conditional check and returns one value if the condition is true and another if false.

Example: `=IF(F1>50, "Pass", "Fail")` checks if the value in cell F1 is greater than 50, and returns "Pass" if true and "Fail" if false.

7. VLOOKUP: Searches for a value in the first column of a table and returns a corresponding value in the same row from another column.

Example: `=VLOOKUP(G1, A1:B10, 2, FALSE)` looks up the value in cell G1 in the table A1 to B10 and returns the value from the second column.

8. HLOOKUP: Similar to VLOOKUP but searches for a value in the first row of a table and returns a corresponding value from another row.

Example: `=HLOOKUP(G2, A1:F1, 2, FALSE)` searches for the value in cell G2 in the first row of the table A1 to F1 and returns the value from the second row.

9. CONCATENATE (or CONCAT): Combines text or values from multiple cells into one cell.

Example: `=CONCATENATE("Hello, ", "World!")` combines the two text strings into "Hello, World!"

10. TEXT: Converts a value to text with a specified number format.

Example: `=TEXT(NOW(), "dd-mmm-yyyy")` formats the current date and time into "02-Nov-2023."

11. LEN: Counts the number of characters in a text string.

Example: `=LEN("Excel")` returns 5 because "Excel" has 5 characters.

12. LEFT: Extracts a specified number of characters from the beginning of a text string.

Example: `=LEFT("Microsoft", 4)` returns "Micro."

13. RIGHT: Extracts a specified number of characters from the end of a text string.

Example: `=RIGHT("Excel", 3)` returns "cell."

14. UPPER and LOWER: Changes the case of text to uppercase or lowercase.

Example: `=UPPER("excel")` returns "EXCEL," and `=LOWER("EXCEL")` returns "excel."

15. DATE: Creates a date by specifying the year, month, and day.

Example: `=DATE(2023, 11, 2)` returns the date "November 2, 2023."

16. TIME: Creates a time by specifying the hour, minute, and second.

Example: `=TIME(15, 30, 0)` returns the time "3:30 PM."

Excel offers a vast library of functions for various tasks, and mastering these basics is essential for efficient spreadsheet use.

SNAKE GAME

What is Snake Game?

Snake is a classic arcade game that has captivated players for decades with its simple yet addictive game play. In the Snake game, players control a snake-like creature that moves around a confined playing area. The objective is to guide the snake to eat food, typically represented by small dots or objects, which causes the snake to grow longer with each meal.

The challenge lies in maneuvering the snake effectively without colliding with walls or the snake's own body. As the snake grows, the player must be increasingly strategic to prevent the snake from running into itself or hitting the boundaries of the playing field.

The game's simplicity is part of its charm, as players only use arrow keys or touch controls to navigate the snake. However, the game play becomes progressively more challenging as the snake's length increases. The more food the snake consumes, the longer it becomes, and the harder it is to avoid collisions. This requires planning routes, quick reflexes, and spatial awareness to maintain the game's momentum.

Snake's appeal is not only due to its game play but also its competitive element. Players aim to achieve high scores by consuming as much food as possible before the snake crashes. The game rewards those who can anticipate the snake's movements and react swiftly to changing circumstances.

Throughout the years, Snake has seen various adaptations and remains a popular choice for both classic gaming enthusiasts and mobile gamers. Its enduring appeal lies in its combination of simple mechanics and challenging game play, making it a timeless favorite among gamers of all ages.

History of Snake Game

The history of the Snake game is a fascinating journey through the world of early video games and mobile gaming. The origins of Snake can be traced back to the late 1970s and early 1980s when it first emerged as a simple yet addictive concept.

Snake's history can be primarily divided into two key phases: the pre-digital era and the digital era.

In the pre-digital era, Snake-like games were often played on early electronic devices and even graphing calculators. The concept was simple: a line or dot that moves around the screen, controlled by arrow keys or buttons, with the objective of "eating" objects that appear on the screen, making the line or dot grow longer. These primitive versions served as precursors to the digital Snake games we know today.

The digital era of Snake began in the late 1970s with arcade machines and early personal computers. An influential version of Snake was included in the arcade game "Blockade," released by Gremlin Industries in 1976. It featured two-player competitive game play, where players aimed to trap their opponent's snake while avoiding collisions themselves.

Snake gained further popularity when it was introduced on the Nokia 6110 mobile phone in 1997. This version was a single-player experience, where players controlled a snake to eat dots while avoiding walls and the snake's tail. The game quickly became a mobile gaming sensation, and its iconic status led to its inclusion in subsequent Nokia phones.

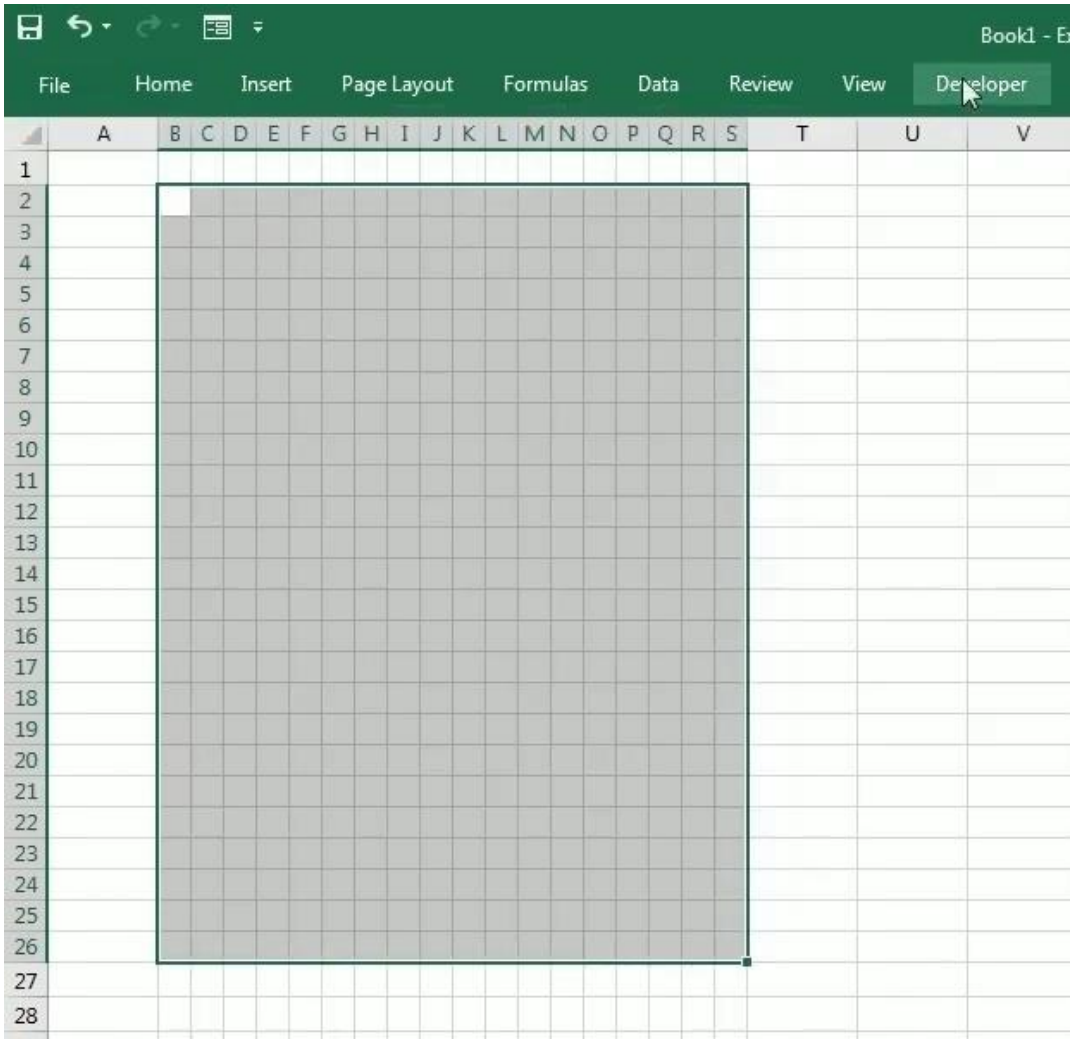
The simplicity and addictiveness of Snake inspired numerous variations and adaptations on different platforms. It has remained a staple in mobile gaming history, showcasing the enduring appeal of straightforward yet challenging game play.

Today, Snake continues to thrive in the digital age, with countless versions and apps available on various devices, preserving the legacy of this iconic game that has evolved and adapted throughout its rich history.

CREATING SNAKE GAME

We can easily create a simple snake game using MS Excel along with using a little bit coding VBA (Visual Basics). Let's see how we can make a snake game in MS Excel.

1. Open MS Excel.
2. Select range of cells on which you want to make snake game board.
   In our case, B2 to S26.
3. Adjust all selected cells to square shape. Like this:



4. Then go to developer tab and select first option to open Visual Basic. Here we are going to code our snake game.

Coding To Start Game and Show Snake on Screen

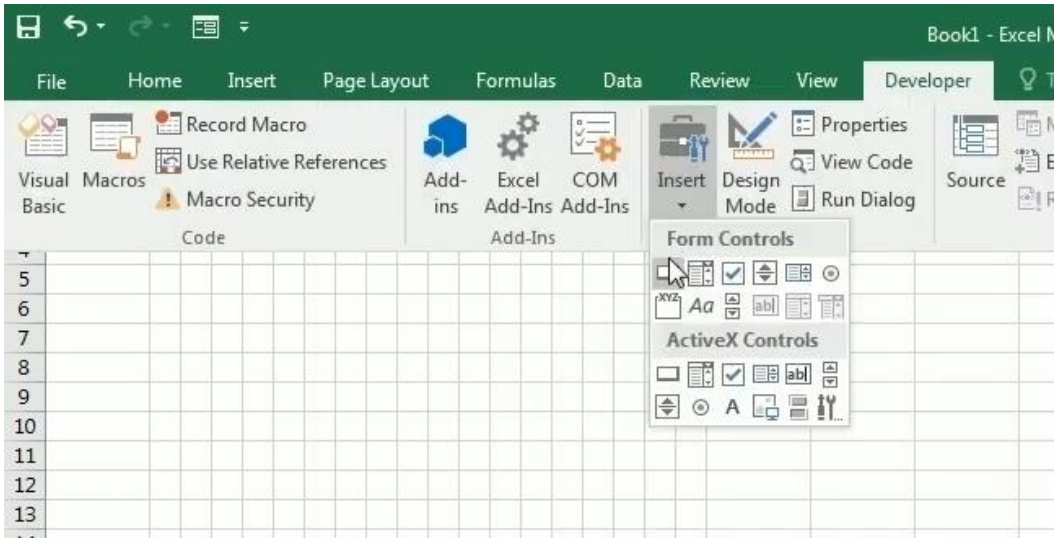
5. In modules, we will create new module from where code reading will begin. So we name it Main.
6. Firstly, we will create two functions in Main StartGame() and ShowSnake to start the game and display a board on screen and a snake on it.
7. We also need to call ShowSnake() inside StartGame() function.
8. Code for StartGame() and ShowSnake functions:

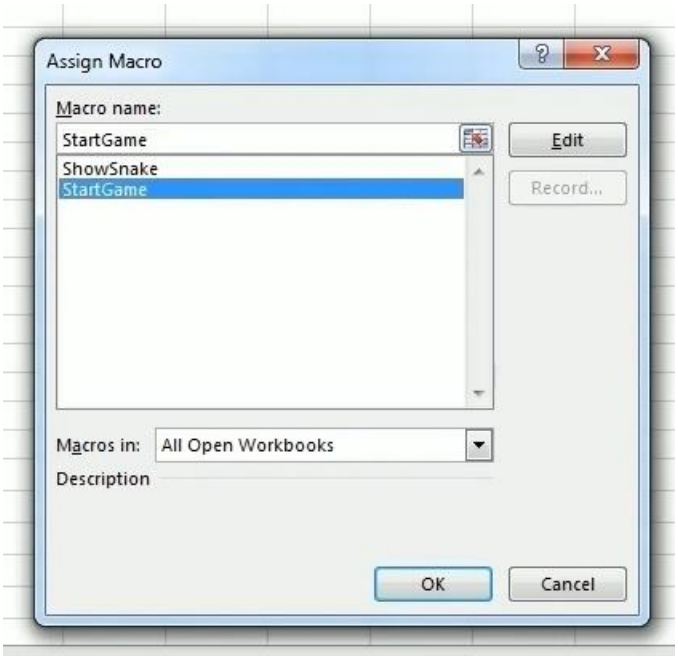
```
Public rinc As Integer, cinc As Integer Dim r() As Integer, c() As
Integer
---------------------------------------------------------------------------
Sub StartGame ()

   Range("B2:S26"). Interior.Color = vbBlack
   ReDim r (2)
   ReDim c (2)
   r(0) = 20: (1) = 21: r(2) = 22
   c(0) = 10: c(1) 10: c (2) = 10
   rinc = 0: cinc = 0
   ShowSnake

End Sub
---------------------------------------------------------------------------
Sub ShowSnake ()


   For i = UBound(r) To 1 Step -1
      Cells (r(i), c(i)). Interior.Color = vbGreen
   Next i
   Cells (r(0), c(0)).Interior.Color = vbRed

End Sub
```

In Main module, we will add just this code for starting the game and show the snake on board.

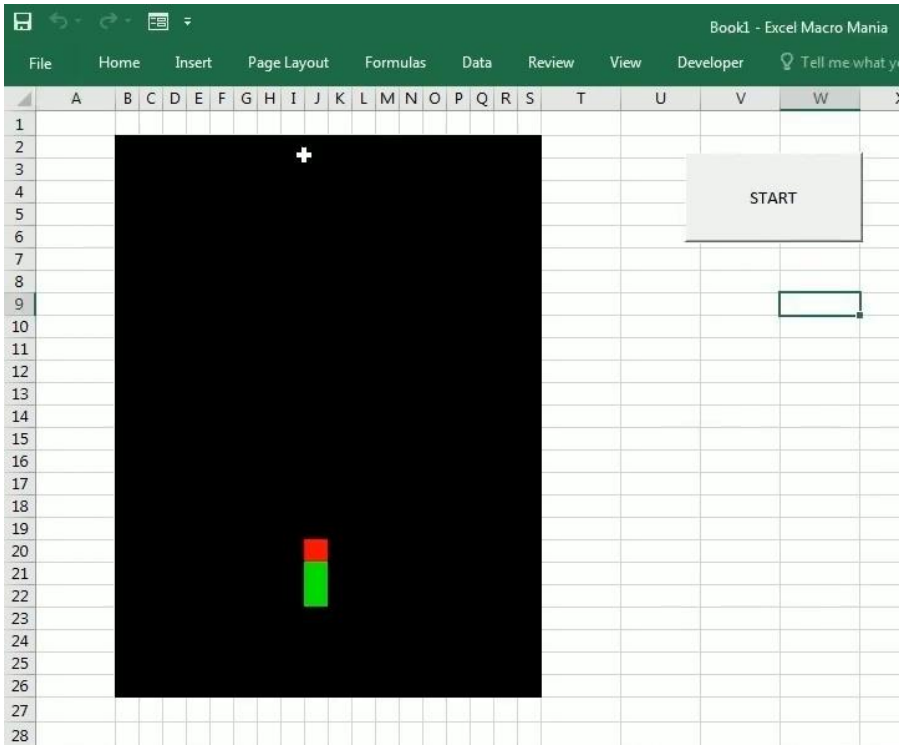
Adding Start Button in Game

9. Then we will come back to Excel screen.
10. Again in developer tab, we will insert a form control button. Attach VBA Macro StartGame and name it as Start.





11. Then if we click on this button, game will get started.
12. A black board with a snake on it will be displayed as output of code written inside Main module.



Setting up Keys for Movement in all 4 directions

13. Then we will make a new module in VBA to set keys for movement for snake and rename the module as Keys.
14. Inside Keys module, we are gonna write code for movement in all 4 directions, i.e. up, right, down, and bottom.
15. Code for setting up arrow keys for movement in all 4 directions:

```
Sub bindKeys ()

    Application. OnKey "(LEFT)", "moveLeft"
    Application. OnKey "(RIGHT)", "moveRigh"
    Application. OnKey "(UP)", "moveUp"
    Application. OnKey "(DOWN)", "moveDown"

End Sub
```

```
Sub moveLeft()

   If cinc <> 1 Then
      cinc = -1
      rinc = 0
      MoveSnake
   End If

End Sub
---------------------------------------------------------------------------------
Sub moveRight ()

   If cinc <> -1 Then
      cinc = 1
      rinc = 0
      MoveSnake
   End If

End Sub
---------------------------------------------------------------------------------
Sub moveUp()

   If rinc <> 1 Then
      cinc = 0
      rinc = -1
      MoveSnake
   End If

End Sub
---------------------------------------------------------------------------------
Sub moveDown ()

   If rinc <> -1 Then
      cinc = 0
      rinc = 1
      MoveSnake
   End If

End Sub
```

```
Sub freeKeys ()

   Application. OnKey "{LEFT}"
   Application. OnKey "{RIGHT}"
   Application. OnKey "{UP}"
   Application. OnKey "{DOWN}"

End Sub
```

Setting movement of snake using keys in game

16. We have just set function for keys of all 4 directions. Now we have to call this function inside Main module where the game starts.
17. To do this, we are gonna add this code to existing code in Main module and call bindKeys function in StartGame code block.

```
Sub MoveSnake ()

   If rinc <> 0 or cinc <> 0 Then
      tail = UBound(r)
      Cells (r(tail), c(tail)).Interior.Color = vbBlack

      'reposition the body of the snake
      For i = tail To 1 Step -1
         r(i)=r(i-1)
         c(i)=c(i-1)
      Next i

      'move the head of the snake
      r(0)=r(0)+r rinc
      c(0) = c(0) + cinc

      ShowSnake
   End If

End Sub
```

18. We have called ShowSnake function in the code also.

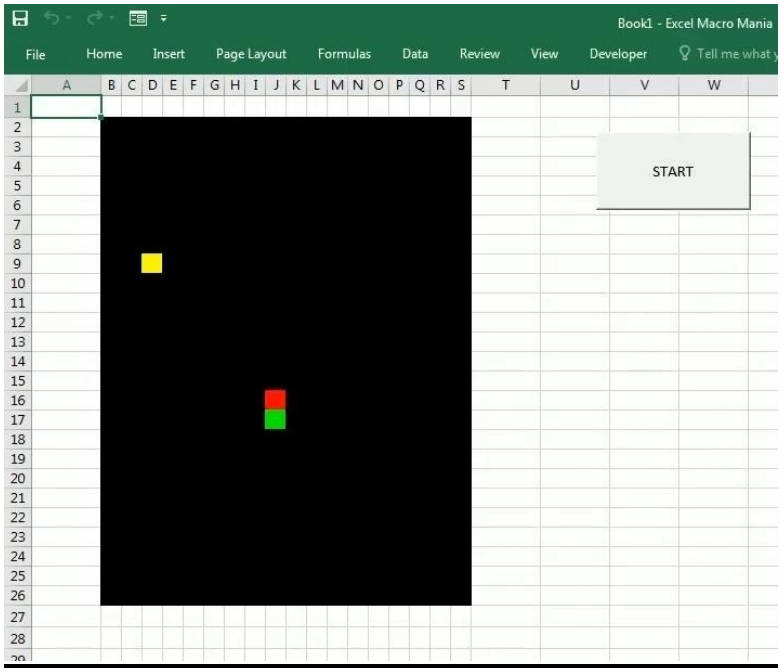
19. Now if we again press start button, snake will begin to move and we can give it direction using arrow keys.

Code to add apple on game board

20. Now we need to throw apple on board. For this, we are gonna write function in Main module to throw apple on board. And we will also call this function AddApple inside StartGame code block.

```
Sub AddApple ()

    Randomize
    arow = Int (Rnd 24) + 2
    acol = Int (Rnd 14) + 2
    Cells (arow, acol). Interior.Color = vbYellow

End Sub
```

21. Now clicking on Start button will let snake move around in board using arrow keys and along with it, a apple will also be added to the board.



Code to check snake movement and increase its length

22. Now we need to check movement of the snake and add condition if snake eats one apple it's length will be increased by one cell and a new Apple will be added to the board. To do this, we are gonna add if else code block inside MoveSnake function.

```
'check the movement of the snake

If Cells (r(0), c(0)). Interior.Color = vbYellow Then

   'got the apple
   apples apples + 1 =
   ReDim Preserve r (UBound(r) + 1)
   ReDim Preserve c(UBound(c) + 1)
   r (UBound(r)) = r (UBound(r) - 1)
   c (UBound(r)) = c(UBound(c) - 1)
   AddApple

ElseIf Cells (r(0), c(0)). Interior.Color <> vbBlack Then

   Exit Sub

End if
```

23. Now if snake eats an apple, it's length will increase by one cell and a new Apple will be thrown on random cell in the board.

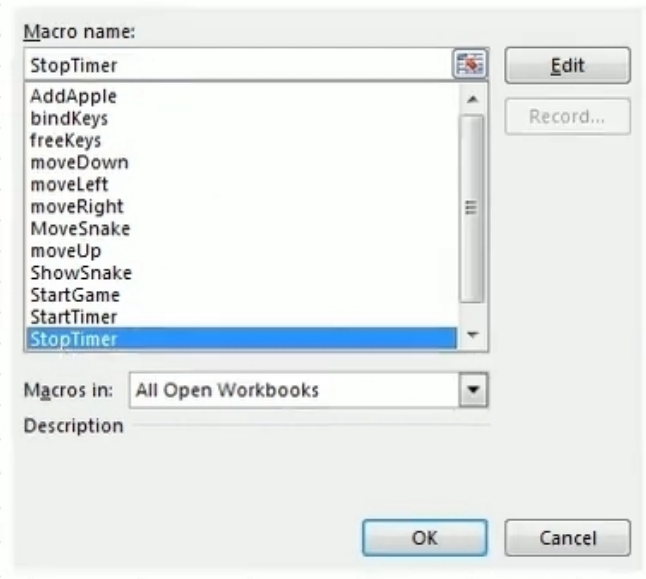
Setting code for ending snake game

24. To stop the game when we want, or when snake hits the edge of the board, or when it hits itself, we need to add timing functions in code and set if else conditions to stop the game if any of above given conditions satisfies.
25. We will make a new module with name Timer to add all timer functions. We need to call StartTimer inside StartGame code block. And call StopTimer inside if else condition set inside MoveSnake code block.

```
Sub StartTimer()

   If TimerID <> 0 Then
      KillTimer 0, TimerID
      TimerID = 0
   End If

   TimerID = SetTimer (0, 0, 200, AddressOf TimerEvent)

End Sub
-------------------------------------------------------------------------------
Sub TimerEvent ()

   On Error Resume Next
   Call MoveSnake

End Sub
-------------------------------------------------------------------------------
Sub StopTimer ()

   KillTimer 0, TimerID
   TimerID = 0
   freeKeys

End Sub
```

Code for calling stop game function in game

26. After this, we need to add oneore button on Excel screen and attach it with StopTimer VBA Macro. We also need to set if else conditions inside MoveSnake code block like this.

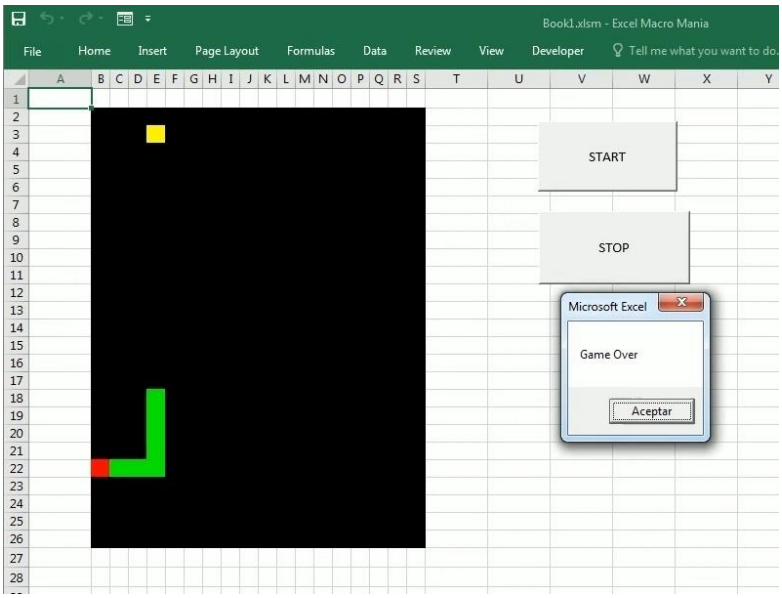



```
'check the movement of the snake

If Cells (r(0), c(0)). Interior.Color = vbYellow Then

   'got the apple
   apples apples + 1 =
   ReDim Preserve r (UBound(r) + 1)
   ReDim Preserve c(UBound(c) + 1)
   r (UBound(r)) = r (UBound(r) - 1)
   c (UBound(r)) = c(UBound(c) - 1)
   AddApple

ElseIf Cells (r(0), c(0)). Interior.Color <> vbBlack Then

   StopTimer
   MsgBox "Game Over"
   Exit Sub

End if
```

27. Inside if else conditions, we have only called StopTimer function and displayed message in message box of "Game Over" inside the Else If code block.

Game Created!!

28. Now if we hit stop button, the game will end and a message box will display the message of Game Over.



29. If snake hits itself, the game will over with the same message box.
30. If snake hits border of the board, the will end with the same message box.
31. Now our game is ready to play.

CONCLUSION

In the course of this project, we successfully designed and implemented a classic Snake game within Microsoft Excel using Visual Basic for Applications (VBA). This undertaking offered several key takeaways:

1. Practical Application of VBA: Developing a game within Excel was an innovative application of VBA. We harnessed the power of VBA to create an interactive user experience in a familiar spreadsheet environment.

2. Coding Proficiency: Through this project, we deepened our knowledge of VBA, honing our skills in coding, debugging, and problem-solving. This provided a valuable hands-on learning experience.

3. User Interface Design: The project allowed us to explore user interface design, where we utilized Excel's built-in features like worksheets, shapes, and form controls to create an intuitive and visually appealing game interface.

4. Game Logic and Algorithms: The game required us to design and implement game logic and algorithms. We learned about concepts such as collision detection, score tracking, and snake movement.

5. Testing and Debugging: Rigorous testing and debugging were essential to ensure the game's functionality and eliminate any potential issues. This emphasized the importance of thorough testing in software development.

6. Performance and Optimization: Excel's limitations in terms of graphics and performance were evident. We had to optimize the code to ensure the game ran smoothly within the Excel environment.

In conclusion, creating a Snake game in MS Excel using VBA was not only a rewarding project but also an educational one. It showcased the versatility of VBA and its potential for creating engaging applications. Moreover, this project served as a platform to enhance our programming skills, user interface design, and problem-solving abilities. It's a testament to what can be achieved by thinking creatively and pushing the boundaries of traditional software applications.