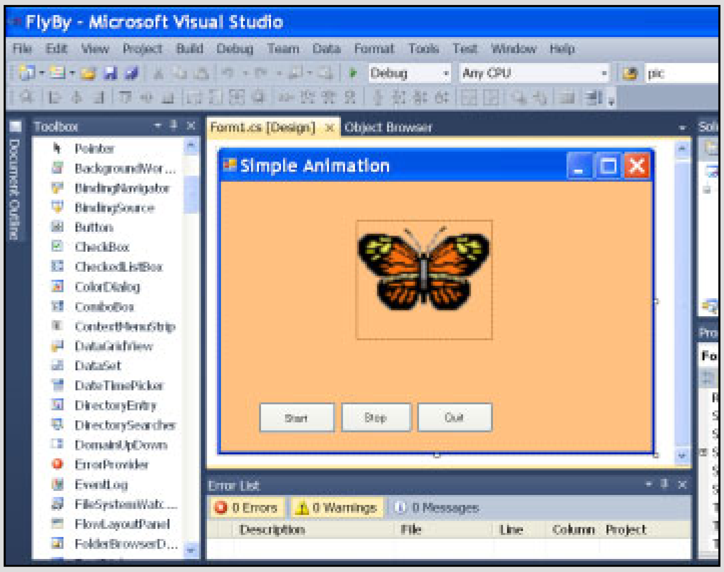
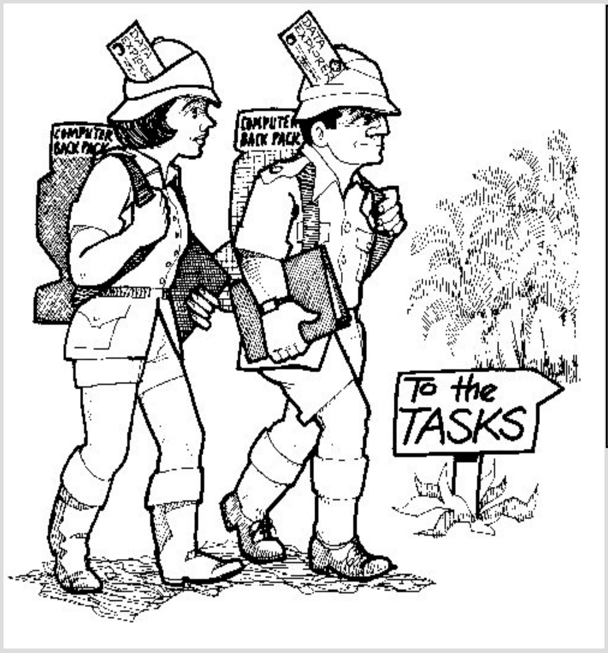
Windows Programming

bucks new university

# Console & Windows Programming Using C#





## Application Programming (CO453)

## Part A Weeks 1-3

# Project Unit: C# Project: The Scissors-Paper-Stone Game

This week we shall begin a slightly larger exercise .. the Scissors-Paper-Stone Game. You should know the rules of the game before you start:

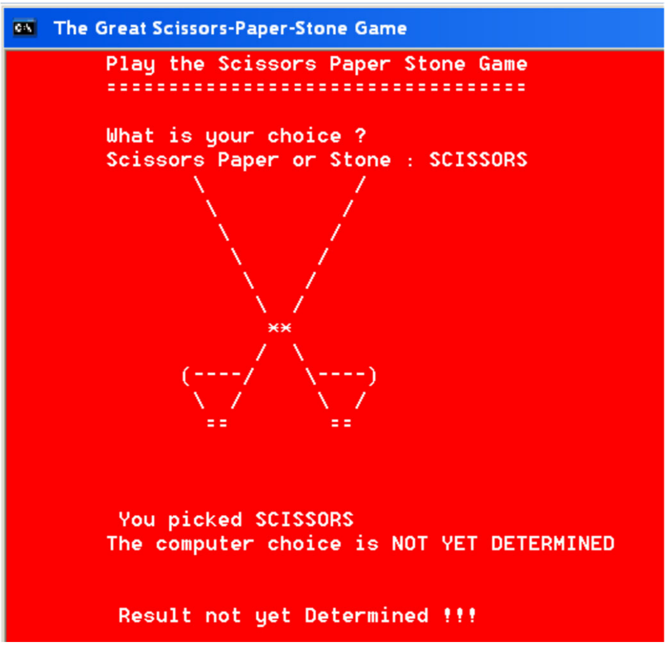
The Basic Rules (playing against the computer)
=====================================

- The player chooses either: Scissors, Paper or Stone
- The computer also chooses one of these at random
- There are various possible results:
  - If player and computer choose the same thing, the result is a Draw.
  - Scissors win against Paper (because scissors cut paper)
  - Scissors lose against Stone (because stone blunts scissors)
  - Paper wins against Stone (because paper wraps round stone)

## Starting the Project

- This project has already been started for you but it is <u>incomplete</u> and needs a lot of work to finish it.
- Start by opening the **SPSProject** and run it .. set the keyboard Caps Lock ON and when you are asked for your choice, type: **SCISSORS**.
- The computer will now make its random choice and you should then see a crude picture of your choice and a result that is either:
  - a <u>DRAW</u> or
  - <u>NOT YET DETERMINED</u>

  (depends what the computer picked)
- Run the program again and try choosing **PAPER** or **STONE**
- Examine the existing code for the program (see later pages)
- Clearly the program is nowhere near finished so you should try adding more code to achieve the following tasks:


The Great Scissors-Paper-Stone Game

```
Play the Scissors Paper Stone Game
==================================

What is your choice ?
Scissors Paper or Stone : SCISSORS
       \            /
        \          /
         \        /
          \      /
           \    /
            \  /
             **
            /  \
      (----/    \----)
       \  /      \  /
        ==        ==



 You picked SCISSORS
The computer choice is NOT YET DETERMINED


 Result not yet Determined !!!
```

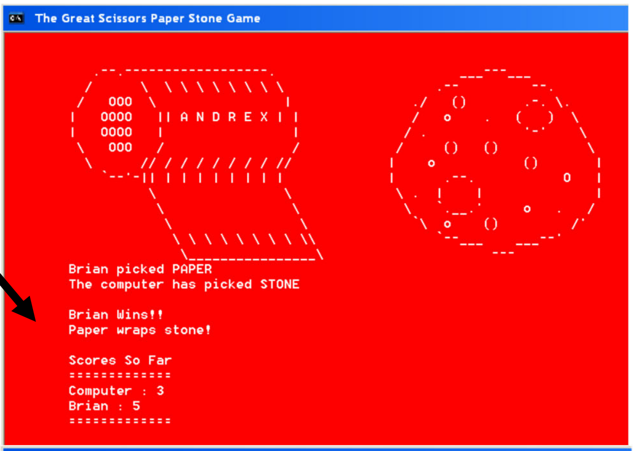
## Basic Project

- Change the background and foreground colours to your own choice.
- Modify the program so that all the computer choices are described correctly, instead of "NOT YET DETERMINED" as above (e.g. The computer chose STONE)
- Get the program to show the result correctly for all possible situations (e.g. THE COMPUTER WINS or YOU WIN) .. instead of "not yet determined" as above.
- Get the program to draw the computer choice as well as the player choice.
- Add a variable for the player name and add code to pick up the name at the start of the program.
- The player name should be used wherever possible e.g

  What is your choice, *Brian*?
  *Brian* picked SCISSORS.  The computer picked PAPER
  *Brian* WON!! Because Scissors Cut Paper.
- Get the program to work for both uppercase and lowercase inputs e.g. it should work if you choose SCISSORS or scissors or Scissors, etc.

## Extension Work 1

- You are to use a scoring system in the game so you must add 2 variables for the ComputerScore and the PlayerScore.
- Implement the scoring as follows:
  - 2 points for a WIN
  - 1 point each for a DRAW
- Add a new method called **showScores()** which prints the scores for the player and computer as shown here:

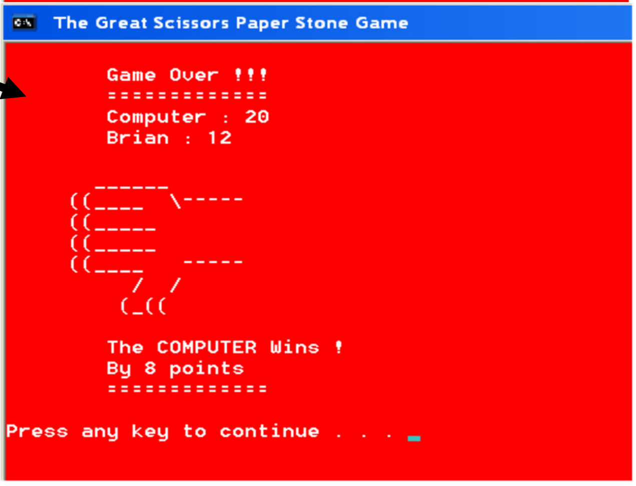


```
Brian picked PAPER
The computer has picked STONE

Brian Wins!!
Paper wraps stone!

Scores So Far
=============
Computer : 3
Brian : 5
=============
```

- Now get the game play to repeat until one of the scores reaches 20.
- Create a new method called **finish()** which is called when the game loop ends.
- The finish() method should clear the screen and then print the results as shown here.
- Use an appropriate picture:
  - ThumbsUp (player win)
  - ThumbsDown (computer win)
  - Smile (draw)
- Note: you will find included some appropriate draw methods for you to use.



```
    Game Over !!!
    =============
    Computer : 20
    Brian : 12


    The COMPUTER Wins !
    By 8 points
    =============

Press any key to continue . . .
```

## Extension Work 2

- Make a copy of your complete Game folder so you don't lose your original game.
- Create a new **Class** in your project and call this **Pictures**
- Remove all the draw methods from your Game class and put them into your Pictures class.
- Get the game to work again using pictures from the Pictures class .. you will have to make several changes e.g. the draw methods should be public instead of private and you will need to create a Pictures object within the Game class.
- Most of the pictures can be positioned anywhere on the screen, but some of them can't .. can you modify these methods so they too can be drawn anywhere?

```
class Game
{
        string compChoice;
        string playerChoice;
        Random randy;

        public static void Main()              // program starts executing here
        {
              Game myGame = new Game ();              // create a new Game object
              myGame.play();                         // call its play method
        }

        public Game()                                 // game constructor
        {
               randy = new Random();                  //  create a new Random object
        }

        public void play()                     // play the game (unfinished)
        {
                     setupScreen();
                     introduction();
                     getPlayerChoice();
                     getComputerChoice();
                     drawPlayerChoice();
                     printChoices();
                     showResult();
                     Console.ReadKey();               // wait for a key press
        }

        private void setupScreen()
        {
                     Console.Title = " The Great Scissors-Paper-Stone Game";
                     Console.SetWindowSize(100, 36);
                     Console.SetBufferSize(100, 36)
                     Console.BackgroundColor = ConsoleColor.Red;
                     Console.ForegroundColor = ConsoleColor.White;
                     Console.Clear();      // clear screen in chosen colour
        }

        private void introduction()
        {
                     Console.WriteLine("\tPlay the Scissors Paper Stone Game");
                     Console.WriteLine("\t==========================");
        }

        private void getPlayerChoice()
        {
                    Console.WriteLine("\n\tWhat is your choice ?");
                    Console.Write("\tScissors Paper or Stone : ");
                    playerChoice = Console.ReadLine();
        }

        // PTO
```

*// Game class continued*

```
private void getComputerChoice()    // unfinished
{
          int num;
          num = randy.Next(3);      // pick a random number (0, 1 or 2)
          if (num == 0)
          {
             compChoice = "SCISSORS";
          }
          else
          {
             compChoice = "NOT YET DETERMINED";
          }
}
```

```
private void drawPlayerChoice()
{
          if (playerChoice == "SCISSORS")
          {
                 drawScissors(10, 5);           // draw Scissors at 10, 5
          }
          else if (playerChoice == "PAPER")
          {
                 drawPaper(10, 5);
          }
          else if (playerChoice == "STONE")
          {
                 drawStone(10, 5);
          }
}
```

```
private void printChoices()
{
          Console.WriteLine("\n\t You picked " + playerChoice);
          Console.WriteLine("\tThe computer has picked " + compChoice);
}
```

```
private void showResult()
{
          if (playerChoice == compChoice)
          {
             Console.WriteLine("\n\tA DRAW!!");
          }
          else
          {
             Console.WriteLine("\n\n\t Result not yet Determined !!!");
          }
}
```

```
private void drawScissors(int x, int y)           // draw at x, y
{
          Console.SetCursorPosition(x, y++);  // set start position then
```

**SPS Game Project Deliverables**
============================

Include the following in your logbook:

- Fully Commented Source Code
- Sample Screen shots
- Completed Test Plan
- Class Diagram(s)
- Commentary on success (or otherwise)

# Some Extra Useful C# Stuff

## Console Screen Instructions

```
Console.BackgroundColor = ConsoleColor.Blue;         // set a background colour
Console.ForegroundColor = ConsoleColor.Yellow; // set a foreground colour
Console.Clear () ;                                   // clear the screen
Console.SetCursorPosition(5, 10);              // x,y position on screen
```

## Pausing the Program

```
Console.ReadKey () ; //  waits for a key to be pressed
```

## Delay for a time

```
System.Threading.Thread.Sleep (1000);
 // gives a delay of 1 second (1000 milliseconds)
```

### The Math.Pow() function

Use the Math.Pow() function to return the power of a number:  e.g.

```
cube = Math.Pow(number, 3);
square = Math.Pow(number, 2);
```

## Converting a string into upper case

```
choice = choice.ToUpper() ;
```

converts the string choice into upper case (e.g. yes becomes YES)

## Random Number Generation

- First we must create a new object from the Random class .. e.g.
  **Random  rand = new  Random();**   *// creates a new object called **rand***
- Then you can use **rand.Next()** to pick the next number e.g.
  *// pick a random number between 0 and 5 and store in an int variable **n***
  *// add 1 to get a number between 1 and 6*
  **n = rand.Next(6) + 1;**  *// puts either 1,2,3,4,5 or 6 into n*
  ( or use **n = rand.Next() % 6 + 1** )

## Output of Decimal Places

- If you want to print a <u>double</u> type of variable (num) to 2 decimal places:
  **Console.WriteLine("The answer is " + num.ToString("0.00"));**

## Alternative Way of Printing Variables

- Instead of :
  **Console.WriteLine("The total of " + n1 + " and " + n2 + " is " + total);**
  You could write:
  **Console.WriteLine("The total of {0} and {1} is {2}",   n1,  n2,  total);**
  *// n1, n2 and total are put in positions {0} {1} and {2} respectively*
- ***or** an alternative if you want 2 decimal places:*
  **Console.WriteLine("The total of {0:F2} and {1:F2} is {2:F2}",   n1,  n2,  total);**
  *// this formats all the numbers to Fixed 2 decimal places*

## Inputting Numbers

- First input into a string variable e.g.
  **input = Console.ReadLine();**
- Then convert this string to the right type and pop into a variable e.g.
  **num = Convert.ToDouble(input);**    or
  **num = Convert.ToInt32(input);**       etc.

# Appendix A: The Basics

## 1. Console Input and Output

**name = Console.ReadLine();** .. store input in a name variable (defined as string)
**Console.WriteLine("I am " + name);** .. output a message with text joined to a name variable
**num1 = Convert.ToDouble ( Console.ReadLine() );** .. enter string and convert to a double
**num2 = Convert.ToInt32 ( Console.ReadLine() );** .. enter string and convert to an integer

## 2. Variables

| | |
|---|---|
| **int count;** | .. define a variable called count to store an integer number |
| **double num;** | .. define a variable called num to store a double (decimal) number |
| **string name;** | .. define a variable called name to store a string (text or words) |

## 3. Assignments to Variables (must be defined first)

| | |
|---|---|
| **count = 0;** | .. put 0 into the count variable (previously defined as int) |
| **num = 5.67;** | .. put 5.67 into the num variable (previously defined as double) |
| **name = "Fred Bloggs";** | .. put these 11 characters in the name variable (defined as string) |

## 4. Calculations

| | |
|---|---|
| **count ++;** | .. add 1 to the value of the count variable |
| **count --;** | .. subtract 1 from the value of the count variable |
| **count = count + 3;** | .. add 3 to value of the count variable (or use **count += 3;** ) |
| **count = count - 6;** | .. subtract 6 from the count variable (or use **count -= 6;** ) |
| **av = (num1 + num2 + num3 + num4) / 4;** | .. work our average of 4 numbers |
| **tax = bill * 17.5 / 100;** | .. work out 17.5 percent tax on your bill |

## 5. Loops (iteration)

### a. The while loop

```
int count = 0;            // initialise a loop counter to zero

while (count < 10)     // continue while loop counter is less than 10
{
        Console.WriteLIne (“The count is " + count);   // repeated
        count ++;       // keep loop going by adding 1 to counter
}
```

### an infinite loop

```
while (true)     // continue the while loop forever
{
        Console.WriteLIne ("Yippeeee!!");   // repeated forever
}
```

### b. The for loop

```
// initialise loop counter; continue while count less than 10 ;  add 1 at end of loop

for (int count = 0; count < 10; count ++)
{
        Console.WriteLine (“The count is " + count);   // repeated 10 times
}
```

### c. The do while loop

```
int  count = 0;            // initialise a loop counter to zero

do
{
        count ++;       // keep loop going by adding 1 to loop counter
        Console.WriteLine ("The count is " + count);   // repeated message
}
while (count < 10);     // continue while loop counter is less than 10
```

## 6. Selection

### a. The if statement

```
        if (count == 4)        // if count is equal to 4
        {
                Console.WriteLine ("We are half way" );
        }
```

### b. The if else statement

```
        if (count >= 4)        // if count is greater or equal to 4
        {
                Console.WriteLine ("We have reached half way" );
        }
        else
        {
                Console.WriteLine ("We are NOT half way yet");
```

### c. The switch statement

```
        switch(count)       // use count value to switch to various cases below:
        {
                case 1:                                    // i.e. if  count value = 1
                        Console.WriteLine ("We are just starting" );            break;
                case 2:  case 3: case 4:
                        Console.WriteLine ("We are on our way" );               break;
                case 4:
                        Console.WriteLine ("We are half way" );                 break;
                default:
                        // do nothing for any other values                      break;
```

## 7. Conditions

| | |
|---|---|
| **(a == b)** | .. a is equal to b ? |
| **(a > b)** | .. a is greater than b ? |
| **(a < b)** | .. a is less than b ? |
| **(a >= b)** | .. a is greater or equal to b ? |
| **(a <= b)** | .. a is less than or equal to b ? |
| **(a != b)** | .. a is NOT equal to b ? |

## 8. Multiple Conditions

```
(a == b  || a == c)              .. a is equal to b OR a is equal to c  ?
(a == b  || a == c || a == d)    .. a is equal to b OR a is equal to c OR a is equal to d ?
(a == b  && a == c)              .. a is equal to b AND a is equal to c  ?
(a <= 100  && a >= 0)            .. a is less than or equal to 100 AND a is greater or equal to 0  ?
```

## 9. Classes, Objects and Methods

```
class Meal                    // define a class called Meal
{
        private string food;  // the class has one class variable (attribute or field)

         public static void Main()            // program starts executing here
         {
               Meal myMeal = new Meal();          // create a new myMeal object
              myMeal.getFood();                   // call the object's getFood() method
         }

        public Meal()                          // this is the Meal class constructor
        {
                food = "Fish and Chips";     // this sets the default food
        }

        public void getFood()           // define a method getFood()which returns nothing
(void)
        {
```

*// this defines a simple class **Meal** which has one variable, one method, one constructor*

## 10. Methods with parameters

- *this defines a method **setTax()** which has 1 parameter (amount) and returns a double value*
- *this method is defined inside a class e.g the Meal class above*
- *to use it, you can 'call' it like this:*
  **vat = *myMeal*.setTax(Bill);** *// assume myMeal is the object created from Meal*

```
public double setTax(double  amount)
{
      double  taxAmount;           // local variable
       taxAmount = amount * 17.5/100;
       return  taxAmount;
}
```

***this passes the value of <u>Bill</u> to the method and picks up a returned tax value from it.***

## 11. try/catch *(simple version : to trap errors or exceptions)*

```
try
{
        // enter instructions to be checked here
}
catch
{
        // error message display here
}
```

## Assessment of CO453 Application Programming

1. The module is assessed by coursework that consists of a series of directed study exercises and programming projects that must be recorded in a logbook.

2. The logbook must be an e-book and should contain your designs, algorithms, test plans, source code and results of your work.

3. The directed study includes **independent study tasks** and **programming projects**.

4. The **classwork** component of the directed study is assessed each week in your practical sessions. You **MUST** be observed doing the classwork in the computer laboratories during these timetabled sessions. You must record your classwork in a logbook and this will be presented for inspection at designated times. Your attendance and achievement will be recorded weekly.

5. The **independent** component of the directed study is your own unaided work. This work must be recorded in your logbook. The **independent** directed study is assessed when logbooks are submitted

6. The **programming projects** are your own unaided work and must be recorded in your logbook. They are assessed at various times during the timetabled practical session.

7. The assessed directed study is contained in several directed study packs.
   The weighting of the assessment to the final grade will be in the order of the following:

**100%: C# Console Independent Study (Units 4 & 5), C# Console Project**

**C# Console Independent Study (Units 4 & 5): 70%**
**C# Console Project: 30%**

## Log Books

For the assignment you should submit (unless otherwise instructed):

- All attempted independent studies and project tasks
- Clearly numbered Task headings
- Simple description of task(s)
- Sample Screen Shots
- Fully commented Source code of relevant sections using highlighter pen to show added code where necessary
- Comments on problems encountered etc.

## Grade related criteria for Programming - CO453

| | |
|---|---|
| **A** | Where the student has demonstrated clear evidence of an excellent understanding of the theories and principles together with a high degree of analytical accuracy, good design skills, implementing fully tested solutions that show reliability, maintainability, readability and minimal complexity and correct form of presentation skills. *To acquire the knowledge and skills to demonstrate the above the student will normally be expected to attend the lecture and practical sessions and attempt at least 85% of independent study for each week.* |
| **B** | Where the student has demonstrated clear evidence of a good understanding of the theories and principles together with a good analytical ability, good design skills, implementing solutions that show reliability, maintainability, readability and minimal complexity and correct form of presentation skills. *To acquire the knowledge and skills to demonstrate the above the student will normally be expected to attend the lecture and practical sessions and attempt at least 75% the independent study for each week.* |
| **C** | Where the student has demonstrated a reasonable understanding of the theories and principles together with a reasonable analytical ability, design skills, implementing solutions that appreciate the need for reliability, maintainability, readability and minimal complexity and reasonable presentation skills. *To acquire the knowledge and skills to demonstrate the above the student will normally be expected to attend the lecture and practical sessions and attempt at least 66% of the independent study for each week.* |
| **D** | Where the student has demonstrated an understanding of the theories and principles of analysis, design, implementation and presentation skills. *To acquire the knowledge and skills to demonstrate the above the student will normally be expected to attend the lecture and practical sessions and attempt at least 50% of the independent study for each week.* |
| **E** | Where the student has made a genuine attempt to acquire the knowledge and skills but requires further application and study to demonstrate an understanding of the theories and principles of analysis, design, implementation and presentation skills. *In order to demonstrate a genuine attempt the student will normally be expected to attend the lecture and practical sessions and attempt at least 40% of the independent study.* |
| **F** | Where the student has clearly not acquired sufficient knowledge and skills and not attempted or coped with the directed study with any degree of competence regarding theories, principles, analysis, design, and implementation and presentation skills<br>or<br>where the student has NOT attended for assessment<br>or<br>where the student has copied work from an alternative source. |

| **Module Name and code** | **Application Programming CO453** |
|---|---|
| **Staff**: Richard Jones, Richard Mather, Carlo Lusuardi & Nick Day | |

**Learning Outcomes:**

- Analyse a simple requirement in a structured manner
- Design, document, implement and test reliable, maintainable programs as solutions to simple problems
- Use structured techniques of design and implementation and good documentation practice.
- Use software development tools.

| Teach WK | Uni WK | LECTURE/TUTORIAL | PRACTICAL |
|---|---|---|---|
| 1 | 18 | **C# (Console) 4 Methods and Parameters** | C# Directed Study Unit 4 |
| 2 | 19 | **C# (Console) 5 Arrays** | C# Directed Study Unit 5 |
| 3 | 20 | **C# (Console) The Project** | C# Directed Study Project Unit |
| 4 | 21 | **Workshop week** | |
| 5 | 22 | **Windows C#1 Introduction & Splash Screen** | C# Directed Study Unit 1 |
| 6 | 23 | **Windows C#2 SPS Game** | C# Directed Study Unit 2 |
| 7 | 24 | **Windows C#3 Other .NET Controls** | C# Directed Study Unit 3 |
| 8 | 25 | **Windows C#4 Multiform projects** | C# Directed Study Unit 4 |
| | **26-28** | **Spring Recess – (Easter)** | |
| 10 | 29 | **Workshop** | |
| 11 | 30 | **Windows C#5 Animation** | C# Directed Study Unit 5 |
| 12 | 31 | **Windows C#6 Graphics** | C# Directed Study Unit 6 |
| 13 | 32 | **Windows C#7 Web Programming** | C# Directed Study Unit 7 |
| 14 | 33 | **Windows C#8 The .Net Project** | C# Directed Study Unit 8 |
| 15 | 34 | **Workshop** | |

**Course Texts:**
Comprehensive Course Notes are provided
Bradley & Millspaugh, ***Programming in C#***, 2010, pub: McGraw Hill
Deitel & Deitel, ***Visual C# 2010 How to Program***, 2011, pub: Pearson