Name: **Session:**
Programming II
Lab Exercise 4.1.2025

In this lab you will create applications to solve the following problems. For each application, turn in your source code as well as a screen shot of your running application.

1. Write an application that displays a Haiku poem. A Haiku poem is written in 3 phrases. The first phrase has five syllables, the second has seven syllables, and the last phrase has five syllables. For example:

   Bright flash then silence
   My expensive computer
   Has gone to heaven

   Your application should define arrays of strings with seven phrases of five syllables and four phrases of seven syllables. Your program should use random number generation to select phrases that follow the 5 – 7 – 5 scheme to generate a random Haiku.

   a. Add the following global variables.
   ```
   string[] five = new string[10];
   string[] seven = new string[5];

   Random r = new Random();
   ```

   b. Add the following code to the Form1_Load event handler.
   ```
   five[0] = "Trying to hold on";
   five[1] = "Nothing stays the same";
   five[2] = "Garden Is dying";
   five[3] = "Garden Is alive";
   five[4] = "Fire ants are stalking";
   five[5] = "My okra looks grand";
   five[6] = "Please open the door";
   five[7] = "Heavy eyelids droop";
   five[8] = "Soft, light cooling breeze";
   five[9] = "for this gift, thank you";

   seven[0] = "So large and incredible!";
   seven[1] = "they found food that's now walking";
   seven[2] = "Easy, fast and light, so light";
   seven[3] = "A garden seeks it's own way";
   seven[4] = "Cool nighttime air drifts in";
   ```

c. Add the following code to the btnGenerate_Click event handler.

```
int rNumber;
string message = "";

rNumber = r.Next(0, 9);
message += five[rNumber] + Environment.NewLine;
rNumber = r.Next(0,4);
message += seven[rNumber] + Environment.NewLine;
rNumber = r.Next(0, 9);
message += five[rNumber];

lblHaiku.Text = message;
```

d. Test your program to endure that it works. Submit a screen shot of your running program.

2. Write an application that generates the child's game "My Grandmothers Trunk". In this game, players sit in a circle and the first player names something that goes in the trunk (i.e. In my grandmothers trunk, I packed a pencil). The next player restates the sentence and adds something new to the trunk: "In my grandmothers trunk, I packed a pencil and a red ball". Each player in turn adds something new to the trunk, attempting to keep track of all the items already there.

   You application should simulate 5 turns in the game. Starting with an empty string, simulate each players turn by concatenating a new word or phrase to the existing string and printing the result.

   a. Add the following global variables.

   ```
   string strMessage = "In my Grandmother's Trunk I packed ";
   bool first = true;
   string test = "aeiouAEIOU";
   int count = 0;
   ```

   b. Add the following code to the Form1_Load event handler.

   ```
   lblBox.Text = strMessage;
   ```

c. Add the following code to the btnAdd_Click event handler.

```
count += 1;

if (first)
{
        //add first item
        if (test.Contains(txtAdd.Text[0]))
                strMessage += " an " + txtAdd.Text;    //word starts with vowel
        else
                strMessage += " a " + txtAdd.Text;     //word starts with consonant
        first = false;
}
else
{
        //add subsequent items
        if (test.Contains(txtAdd.Text[0]))
                strMessage += " and an " + txtAdd.Text;     //word starts with vowel
        else
                strMessage += " and a " + txtAdd.Text;     //word starts with consonant
}

lblBox.Text = strMessage;
txtAdd.Text = "";
txtAdd.Focus();

if (count == 5)
{
        lblBox.Text += ".";
        btnAdd.Enabled = false;
        txtAdd.Enabled = false;
}
```

d. Add the following code to the txtAdd_KeyDown event handler.

```
if (e.KeyCode == Keys.Enter)
{
        //Keep track of the number of items added to the trunk
        count += 1;
        if (first)
        {
                //add first item
                if (test.Contains(txtAdd.Text[0]))
                        strMessage += " an " + txtAdd.Text; //word starts with vowel
                else
                        strMessage += " a " + txtAdd.Text;   //starts with consonant
                first = false;
        }
        else
        {
                //add subsequent items
                if (test.Contains(txtAdd.Text[0]))
                   strMessage += " and an " + txtAdd.Text;   //starts with vowel
                else
                   strMessage += " and a " + txtAdd.Text;     //starts with consonant
        }

        //Add message to textbox
        lblBox.Text = strMessage;
        txtAdd.Text = "";
        txtAdd.Focus();

        //Finished adding items
        if (count == 5)
        {
                lblBox.Text += ".";
                btnAdd.Enabled = false;
                txtAdd.Enabled = false;
        }
}
```

e. Test your program. When it is working paste a screen shot into a word processing document and submit it.

In this activity you may create either a Console Application or a Windows Form Application. When you have completed these applications, submit your source code as well as a screen shot of your running application.

3. The Wicks Corporation manufactures a line of 14 different waxes for cross-country skiing. Because many of their customers have a difficult time selecting the proper wax to use, the company has decided to sell a hand-held computer to aid in the selection. You have been hired to write the program for this computer. Wax choice depends on temperature and snow conditions. The waxes come in various degrees of hardness that are divided into six color groups. A skier selects a wax color on the basis of temperature. All of the color groups except Yellow and White have three varieties (Special, Standard, and Extra) to account for variations in snow conditions (Powder, Firm, and Crusty). Your program should accept as input the current temperature and snow condition. Then compute and print out the most appropriate wax. The temperature and snow condition guidelines are shown below.

Temperature Guidelines (used to select a wax group)

Wax Group	*Temperature*
Yellow	38 < Temp
Red	31 < Temp <= 38
Violet	26 < Temp <= 31
Blue	18 < Temp <= 26
Green	5 < Temp <= 18
White	Temp <= 5

Snow Condition Guidelines (used to select a variety for waxes other than the extreme - temperature waxes (Yellow and White).

Wax Variety	*Snow Conditions*
Special	Powder
Standard	Firm
Extra	Crusty

Inputs

Temperature, the current temperature, an integer.
Valid range: -50 < Temperature < 100

Snow, the current snow condition.
Powder, Firm, Crusty

Outputs

Temperature and snow condition.
The wax variety.
The wax color.
An error message if the input data is invalid.

Functions

Get the temperature and snow condition.

Determine wax color (given the temperature) using the temperature guidelines table above.

Determine was variety (given the snow conditions) using the snow conditions guidelines table above.

Print the wax variety and the wax color.

Print "Invalid input" if either the temperature or snow conditions are invalid.

Sample Execution:

```
Enter the current temperature: 35
Enter the snow conditions (P=powder, F=firm, C=Crusty): P
The best wax is: Special Red
```

4. Old bones and artifacts can be dated using a technique known as radioactive dating. When an artifact is created, it contains some amount of material (e.g., carbon 14) which emits radioactive particles. Over time, the radioactive substance decays and emits particles more and more slowly. The amount of radioactive material never reaches zero, it just reduces geometrically. The amount of time it takes for the rate of radioactive particle emissions to cut in half is constant and is known as the half-life. By knowing the half-life and the initial emissions rate, and measuring the current emissions rate, the age of an artifact can be computed. For example, if the half-life of the radioactive substance is 3 years and the emissions rate was initially 16 and is now 4, then the emissions rate has been cut in half twice, so the artifact must be two half-lives old, or 6 years old.

Equation for calculating age

$$age = \log_2\left(\frac{initialCount}{finalCount}\right) * halfLife$$

Input Format

Each line of input contains three real numbers, h, i and c describing an artifact.
h > 0 is the known half-life of the radioactive substance in years, i > 0 is the known initial emissions rate, and 0 < c ≤ i is the measured current emissions rate.

Output Format

For each artifact, compute and output the age of the artifact in years.

Input Sample

```
3.0  16.0  4.0
25.0  1000.0  1.0
10.0  58.0  0.01
1.0  8.0  1.0
1000.0  25500.0  0.00002
```

Output Sample

```
Age: 6.00 years
Age: 249.14 years
Age: 125.02 years
Age: 3.00 years
Age: 30247.85 years
```