Lab 5: Subroutine and Functions Lab

Problem-1 (HW -1): Working with subroutines

Objectives: Design a VB program to compute an integral of a function using the trapezoidal rule and using a subroutine.

Three circular subdivisions need to be cleared for construction. The XYZ tree cutting company charges $100 per square feet to clear trees and to stabilize the area, and would charge $5 per linear ft to install fence in that area. Approximate radii of the sites are estimated to be 50, 75, and 100 ft. The program below does most of the work for you, but you need modify it slightly to evaluate the costs.

<table>
<thead>
<tr>
<th>No of plots</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Radius of the plots</td>
<td>50</td>
</tr>
<tr>
<td>Total area</td>
<td>56964.28125</td>
</tr>
<tr>
<td>Total fence length</td>
<td>1414.285645</td>
</tr>
<tr>
<td>Total clearing cost</td>
<td>??</td>
</tr>
<tr>
<td>Fencing cost</td>
<td>???</td>
</tr>
</tbody>
</table>

Private Sub CommandButton1_Click()
Dim n As Integer
Dim r1 As Single, a1 As Single, p1 As Single
Dim r2 As Single, a2 As Single, p2 As Single, total_area As Single
Dim r3 As Single, a3 As Single, p3 As Single, total_perimeter As Single
n = Cells(1, 2)
r1 = Cells(2, 2)
r2 = Cells(2, 3)
r3 = Cells(2, 4)
Call mycircle(r1, a1, p1)
Call mycircle(r2, a2, p2)
Call mycircle(r3, a3, p3)
total_area = a1 + a2 + a3
total_perimeter = p1 + p2 + p3
Cells(3, 2) = total_area
Cells(4, 2) = total_perimeter
End Sub

Sub mycircle(r As Single, a As Single, p As Single)
Dim pi As Single
pi = 22# / 7#
a = pi * (r ^ 2)
p = 2# * pi * r
End Sub

Now let us look into the program and learn exactly how subroutines work! The following lines are the key new lines where subroutines are “called” or invoked by the main program:

*Call mycircle(r1, a1, p1)*
*Call mycircle(r2, a2, p2)*
*Call mycircle(r3, a3, p3)*

Here you call the subroutine, mycircle, to compute the area and the perimeter of the sites. Note you “pass” the values of r1 into these routines and the routine computes the results and sends them back to your main program as a1 and p1. It is good to use a convention where all the input variables are listed first.
followed by the list of the output variables. Similarly, in the next two lines, the radii r2 and r3 are passed into the subroutine to evaluate a2, p2, a3 and p3.

\[
\text{total\_area} = a_1 + a_2 + a_3 \\
\text{total\_perimeter} = p_1 + p_2 + p_3
\]

This computes the total area and perimeter of the sites by summing the individual site areas and perimeters.

\[
\text{Cells}(3, 2) = \text{total\_area} \\
\text{Cells}(4, 2) = \text{total\_perimeter}
\]

This outputs the variables total\_area and total\_perimeter onto the spreadsheet.

**Sub mycircle(r As Single, a As Single, p As Single)**

\[
\begin{align*}
\text{Dim pi As Single} \\
\pi &= \frac{22\#}{7\#} \\
a &= \pi * (r ^ 2) \\
p &= 2\# * \pi * r
\end{align*}
\]

*End Sub*

The code above is a subroutine that computes the area and perimeter of site with radius “r”. The area of the site is stored in the variable “a” whereas the perimeter of the site is stored in the variable “p”. This routine is invoked when “Call mycircle(r1, a1, p1)” is executed in the main program. Note, the value of r1 is transferred (or mapped) into the subroutine as r, the value in the variables “a” and “p” are computed within the subroutine and are transferred (or mapped) back to the main program as the variables “a1” and “p1”.

Visual basic, by default, will pass the variable values into a subroutine “by reference.” Another option to pass values is “by value.” The differences are explained in your lectures notes. When the variable names are passed by reference, the computer will use the original memory locations that have already been setup for the variables in the calling routine. The mapped variables simply refer to the same memory location, even if their names are different. One possible limitation of this strategy is that the local routine may inadvertently change the value of the variable that is not supposed to be changed. For example, if the value of n is set to, say 100, either in the routine circle or square then the value of n will become 100 in the main routine too. This type of situation can lead to serious run time bugs that are difficult to track. To avoid such problems, passing “by value” method may be used. However, we will only use pass by reference in this class.

**Problem – 2 (HW-2):** Develop a VB program to compute \[ I = \int_{1}^{5} x^2 \ln(x) dx \] using trap rule. Use \( n = 10 \).

Refer to HW for pseudo code and formula.

**Problem – 3 (HW-6):** Develop a VB program to compute \[ I = \int_{1}^{5} x^2 \ln(x) dx \] using Simpson’s rule using \( n = 10 \). Use the pseudo code and formula given in your homework.

**Problem -4 (HW-3):** Modify the subroutine in problem – 2 (trap rule problem) and use a function to evaluate the integral. Refer to class notes (ppt slides) for details on how to use functions.

**Problem 5:** Do HW problem #5