**CO1109 CW1**

**Jack Hawkins**

Code:

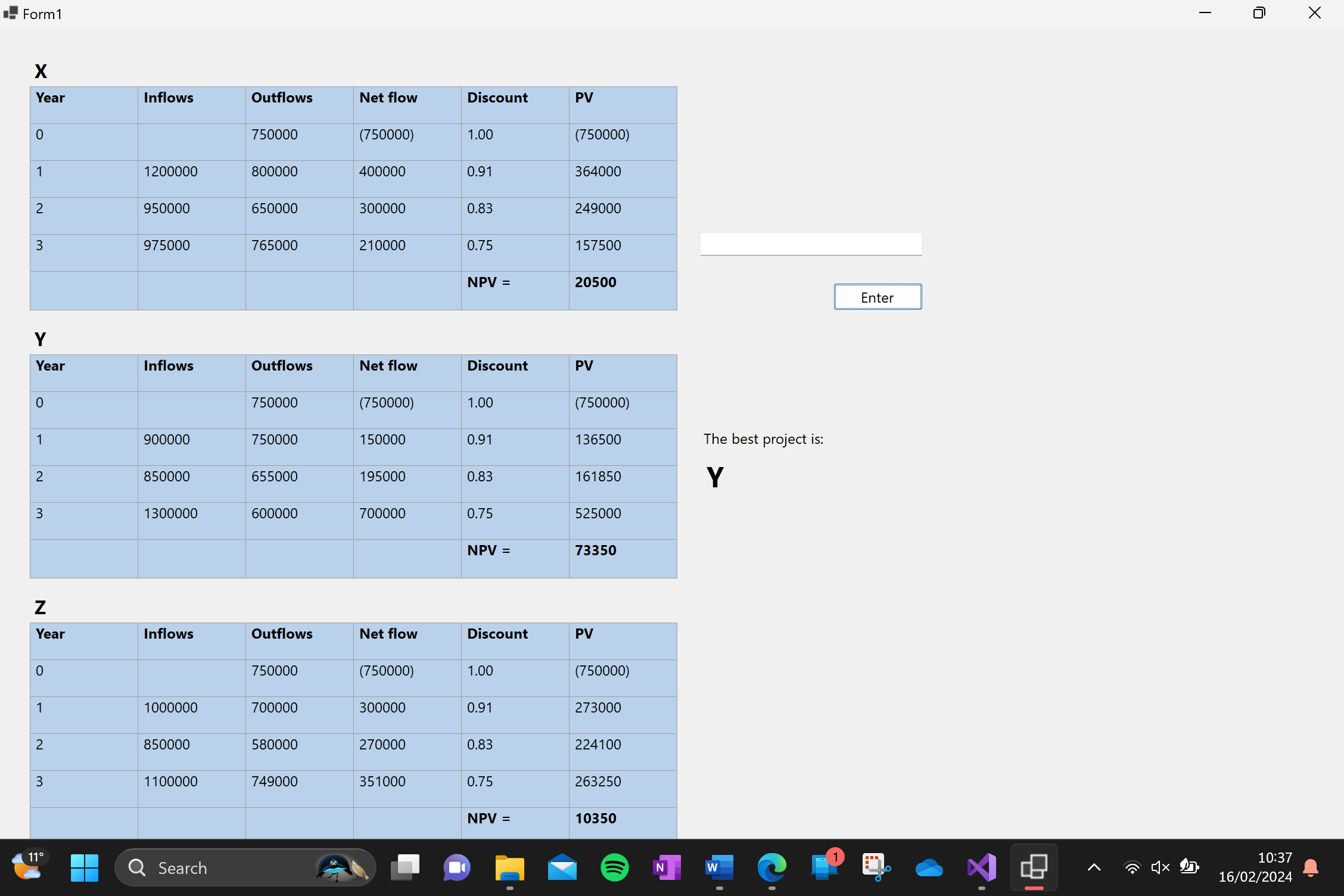
|  |
| --- |
| using System;  using System.Windows.Forms;  namespace CO1109\_CW1  {  public partial class Form1 : Form  {  public Form1()  {  InitializeComponent();  }  public int count = 0;  public string projectXName = "";  public string projectYName = "";  public string projectZName = "";  public double netFlowX = 0;  public double netFlowY = 0;  public double netFlowZ = 0;  public double xNPV = 0;  public double yNPV = 0;  public double zNPV = 0;  public void UpdateLabel(Label label, string text)  {  // This method updates the passed label with the text passed and ensures that the label is visible.  if (text[0] == '-') // If the text begins with '-', remove it and place the string in brackets.  {  text = "(" + text.Remove(0, 1) + ")";  }  if (decimal.TryParse(text, out \_)) // If the text can be converted to a decimal, round to 2 decimal places.  {  text = Convert.ToString(Math.Round(Convert.ToDecimal(text), 2));  }  label.Text = text;  if (!label.Visible)  {  label.Visible = true;  }  }  private void EnterButton\_Click(object sender, EventArgs e)  {  // The program operates using this method which activates when the enter button is clicked.  // It utilises a switch-case statement to determine what happens each time the button is clicked.  // The switch depends on the variable count, which is incremented after each valid input.  // Invalid inputs are: an empty string, non-number values after the project names have been entered.  // If the user enters an invalid input, a message will be displayed and the count value will not be incremented.    if (InputTextBox.Text != "") // If the input is not empty, go to the switch statement.  {  // The below switch statement sets the names for the projects for the first three valid inputs.  // Each following valid input allows the user to enter a value into the table.  // Each case calls the UpdateLabel method, which takes a label and the user's input as its arguments.  // Each case also updates the prompt ready for the following input case.  switch (count)  {  case 0: // Project X name  UpdateLabel(ProjectXLabel, InputTextBox.Text);  projectXName = InputTextBox.Text;  PromptLabel.Text = "Please enter a name for Project Y";  count++;  break;  case 1: // Project Y name  UpdateLabel(ProjectYLabel, InputTextBox.Text);  projectYName = InputTextBox.Text;  PromptLabel.Text = "Please enter a name for Project Z";  count++;  break;  case 2: // Project Z name  UpdateLabel(ProjectZLabel, InputTextBox.Text);  projectZName = InputTextBox.Text;  PromptLabel.Text = "Please enter the year 0 outflow for " + projectXName;  count++;  break;  case 3: // Outflow 0 X  if (int.TryParse(InputTextBox.Text, out int \_))  {  UpdateLabel(Outflow0X, InputTextBox.Text);  PromptLabel.Text = "Please enter the year 1 inflow for " + projectXName;  netFlowX -= Convert.ToDouble(InputTextBox.Text);  UpdateLabel(NetFlow0X, Convert.ToString(netFlowX));  double pv = netFlowX;  UpdateLabel(PV0X, Convert.ToString(pv));  xNPV += pv;  count++;  }  else  {  MessageBox.Show("Value must be an integer.");  }  break;  case 4: // Inflow 1 X  if (int.TryParse(InputTextBox.Text, out \_))  {  UpdateLabel(Inflow1X, InputTextBox.Text);  PromptLabel.Text = "Please enter the year 1 outflow for " + projectXName;  netFlowX = Convert.ToDouble(InputTextBox.Text);  count++;  }  else  {  MessageBox.Show("Value must be an integer.");  }  break;  case 5: // Outflow 1 X  if (int.TryParse(InputTextBox.Text, out \_))  {  UpdateLabel(Outflow1X, InputTextBox.Text);  PromptLabel.Text = "Please enter the year 2 inflow for " + projectXName;  netFlowX -= Convert.ToDouble(InputTextBox.Text);  UpdateLabel(NetFlow1X, Convert.ToString(netFlowX));  double pv = netFlowX \* 0.91;  UpdateLabel(PV1X, Convert.ToString(pv));  xNPV += pv;  count++;  }  else  {  MessageBox.Show("Value must be an integer.");  }  break;  case 6: // Inflow 2 X  if (int.TryParse(InputTextBox.Text, out \_))  {  UpdateLabel(Inflow2X, InputTextBox.Text);  PromptLabel.Text = "Please enter the year 2 outflow for " + projectXName;  netFlowX = Convert.ToDouble(InputTextBox.Text);  count++;  }  else  {  MessageBox.Show("Value must be an integer.");  }  break;  case 7: // Outflow 2 X  if (int.TryParse(InputTextBox.Text, out \_))  {  UpdateLabel(Outflow2X, InputTextBox.Text);  PromptLabel.Text = "Please enter the year 3 inflow for " + projectXName;  netFlowX -= Convert.ToDouble(InputTextBox.Text);  UpdateLabel(NetFlow2X, Convert.ToString(netFlowX));  double pv = netFlowX \* 0.83;  UpdateLabel(PV2X, Convert.ToString(pv));  xNPV += pv;  count++;  }  else  {  MessageBox.Show("Value must be an integer.");  }  break;  case 8: // Inflow 3 X  if (int.TryParse(InputTextBox.Text, out \_))  {  UpdateLabel(Inflow3X, InputTextBox.Text);  PromptLabel.Text = "Please enter the year 3 outflow for " + projectXName;  netFlowX = Convert.ToDouble(InputTextBox.Text);  count++;  }  else  {  MessageBox.Show("Value must be an integer.");  }  break;  case 9: // Outflow 3 X  if (int.TryParse(InputTextBox.Text, out \_))  {  UpdateLabel(Outflow3X, InputTextBox.Text);  PromptLabel.Text = "Please enter the year 0 outflow for " + projectYName;  netFlowX -= Convert.ToDouble(InputTextBox.Text);  UpdateLabel(NetFlow3X, Convert.ToString(netFlowX));  double pv = netFlowX \* 0.75;  UpdateLabel(PV3X, Convert.ToString(pv));  xNPV += pv;  UpdateLabel(NPVValueX, Convert.ToString(xNPV));  count++;  }  else  {  MessageBox.Show("Value must be a number.");  }  break;  case 10: // Outflow 0 Y  if (int.TryParse(InputTextBox.Text, out \_))  {  UpdateLabel(Outflow0Y, InputTextBox.Text);  PromptLabel.Text = "Please enter the year 1 inflow for " + projectYName;  netFlowY -= Convert.ToDouble(InputTextBox.Text);  UpdateLabel(NetFlow0Y, Convert.ToString(netFlowY));  double pv = netFlowY;  UpdateLabel(PV0Y, Convert.ToString(pv));  yNPV += pv;  count++;  }  else  {  MessageBox.Show("Value must be an integer.");  }  break;  case 11: // Inflow 1 Y  if (int.TryParse(InputTextBox.Text, out \_))  {  UpdateLabel(Inflow1Y, InputTextBox.Text);  PromptLabel.Text = "Please enter the year 1 outflow for " + projectYName;  netFlowY = Convert.ToDouble(InputTextBox.Text);  count++;  }  else  {  MessageBox.Show("Value must be an integer.");  }  break;  case 12: // Outflow 1 Y  if (int.TryParse(InputTextBox.Text, out \_))  {  UpdateLabel(Outflow1Y, InputTextBox.Text);  PromptLabel.Text = "Please enter the year 2 inflow for " + projectYName;  netFlowY -= Convert.ToDouble(InputTextBox.Text);  UpdateLabel(NetFlow1Y, Convert.ToString(netFlowY));  double pv = netFlowY \* 0.91;  UpdateLabel(PV1Y, Convert.ToString(pv));  yNPV += pv;  count++;  }  else  {  MessageBox.Show("Value must be an integer.");  }  break;  case 13: // Inflow 2 Y  if (int.TryParse(InputTextBox.Text, out \_))  {  UpdateLabel(Inflow2Y, InputTextBox.Text);  PromptLabel.Text = "Please enter the year 2 outflow for " + projectYName;  netFlowY = Convert.ToDouble(InputTextBox.Text);  count++;  }  else  {  MessageBox.Show("Value must be an integer.");  }  break;  case 14: // Outflow 2 Y  if (int.TryParse(InputTextBox.Text, out \_))  {  UpdateLabel(Outflow2Y, InputTextBox.Text);  PromptLabel.Text = "Please enter the year 3 inflow for " + projectYName;  netFlowY -= Convert.ToDouble(InputTextBox.Text);  UpdateLabel(NetFlow2Y, Convert.ToString(netFlowY));  double pv = netFlowY \* 0.83;  UpdateLabel(PV2Y, Convert.ToString(pv));  yNPV += pv;  count++;  }  else  {  MessageBox.Show("Value must be an integer.");  }  break;  case 15: // Inflow 3 Y  if (int.TryParse(InputTextBox.Text, out \_))  {  UpdateLabel(Inflow3Y, InputTextBox.Text);  PromptLabel.Text = "Please enter the year 3 outflow for " + projectYName;  netFlowY = Convert.ToDouble(InputTextBox.Text);  count++;  }  else  {  MessageBox.Show("Value must be an integer.");  }  break;  case 16: // Outflow 3 Y  if (int.TryParse(InputTextBox.Text, out \_))  {  UpdateLabel(Outflow3Y, InputTextBox.Text);  PromptLabel.Text = "Please enter the year 0 outflow for " + projectZName;  netFlowY -= Convert.ToDouble(InputTextBox.Text);  UpdateLabel(NetFlow3Y, Convert.ToString(netFlowY));  double pv = netFlowY \* 0.75;  UpdateLabel(PV3Y, Convert.ToString(pv));  yNPV += pv;  UpdateLabel(NPVValueY, Convert.ToString(yNPV));  count++;  }  else  {  MessageBox.Show("Value must be a number.");  }  break;  case 17: // Outflow 0 Z  if (int.TryParse(InputTextBox.Text, out \_))  {  UpdateLabel(Outflow0Z, InputTextBox.Text);  PromptLabel.Text = "Please enter the year 1 inflow for " + projectZName;  netFlowZ -= Convert.ToDouble(InputTextBox.Text);  UpdateLabel(NetFlow0Z, Convert.ToString(netFlowZ));  double pv = netFlowZ;  UpdateLabel(PV0Z, Convert.ToString(pv));  zNPV += pv;  count++;  }  else  {  MessageBox.Show("Value must be an integer.");  }  break;  case 18: // Inflow 1 Z  if (int.TryParse(InputTextBox.Text, out \_))  {  UpdateLabel(Inflow1Z, InputTextBox.Text);  PromptLabel.Text = "Please enter the year 1 outflow for " + projectZName;  netFlowZ = Convert.ToDouble(InputTextBox.Text);  count++;  }  else  {  MessageBox.Show("Value must be an integer.");  }  break;  case 19: // Outflow 1 Z  if (int.TryParse(InputTextBox.Text, out \_))  {  UpdateLabel(Outflow1Z, InputTextBox.Text);  PromptLabel.Text = "Please enter the year 2 inflow for " + projectZName;  netFlowZ -= Convert.ToDouble(InputTextBox.Text);  UpdateLabel(NetFlow1Z, Convert.ToString(netFlowZ));  double pv = netFlowZ \* 0.91;  UpdateLabel(PV1Z, Convert.ToString(pv));  zNPV += pv;  count++;  }  else  {  MessageBox.Show("Value must be an integer.");  }  break;  case 20: // Inflow 2 Z  if (int.TryParse(InputTextBox.Text, out \_))  {  UpdateLabel(Inflow2Z, InputTextBox.Text);  PromptLabel.Text = "Please enter the year 2 outflow for " + projectZName;  netFlowZ = Convert.ToDouble(InputTextBox.Text);  count++;  }  else  {  MessageBox.Show("Value must be an integer.");  }  break;  case 21: // Outflow 2 Z  if (int.TryParse(InputTextBox.Text, out \_))  {  UpdateLabel(Outflow2Z, InputTextBox.Text);  PromptLabel.Text = "Please enter the year 3 inflow for " + projectZName;  netFlowZ -= Convert.ToDouble(InputTextBox.Text);  UpdateLabel(NetFlow2Z, Convert.ToString(netFlowZ));  double pv = netFlowZ \* 0.83;  UpdateLabel(PV2Z, Convert.ToString(pv));  zNPV += pv;  count++;  }  else  {  MessageBox.Show("Value must be an integer.");  }  break;  case 22: // Inflow 3 Z  if (int.TryParse(InputTextBox.Text, out \_))  {  UpdateLabel(Inflow3Z, InputTextBox.Text);  PromptLabel.Text = "Please enter the year 3 outflow for " + projectZName;  netFlowZ = Convert.ToDouble(InputTextBox.Text);  count++;  }  else  {  MessageBox.Show("Value must be an integer.");  }  break;  case 23: // Outflow 3 Z  if (int.TryParse(InputTextBox.Text, out \_))  {  UpdateLabel(Outflow3Z, InputTextBox.Text);  PromptLabel.Visible = false;  netFlowZ -= Convert.ToDouble(InputTextBox.Text);  UpdateLabel(NetFlow3Z, Convert.ToString(netFlowZ));  double pv = netFlowZ \* 0.75;  UpdateLabel(PV3Z, Convert.ToString(pv));  zNPV += pv;  UpdateLabel(NPVValueZ, Convert.ToString(zNPV));  count++;  double max = xNPV;  char maximum = 'x';  // This finds the highest NPV value and sets the variable maximum to the corresponding letter.  if (yNPV > max)  {  max = yNPV;  maximum = 'y';  }  if (zNPV > max)  {  max = zNPV;  maximum = 'z';  }  // This updates the result label with the correct name.  switch (maximum)  {  case 'x':  UpdateLabel(ResultLabel, projectXName);  break;  case 'y':  UpdateLabel(ResultLabel, projectYName);  break;  case 'z':  UpdateLabel(ResultLabel, projectZName);  break;  default:  break;  }  }  else  {  MessageBox.Show("Value must be a number.");  }  break;  default: // When count > 23, i.e., the analysis is complete.  MessageBox.Show("NPV analysis complete - restart program to analyse again.");  break;  }  // This clears the input box after each input so that the user does not accidentally enter the same value again.  InputTextBox.Text = "";  }  else // If the input is empty, tell the user it cannot be empty.  {  MessageBox.Show("Input cannot be empty.");  }  }  private void Form1\_Load(object sender, EventArgs e)  {  WindowState = FormWindowState.Maximized; // This sets the window to full screen when it loads.  }  }  } |

Initial form:

A screenshot of a computer

Description automatically generated

Final result:



Narrative:

1. The program asks the user to enter the names for each project and updates the names on the form.
2. The program then asks the user for the data for project X.
   1. It asks for the initial outflow and displays it in the year 0 outflow column and the year 0 net flow column, as well as the present value column.
   2. It then asks for the inflow for year 1 and then the outflow for year 1 and uses these to calculate the net flow for year 1. The net flow is then multiplied by 0.91 to get the present value for year 1.
   3. The above step is repeated for years 2 and 3, multiplying the net flow by 0.83 and 0.75 respectively.
   4. Once all three present value figures have been calculated, they are added together and displayed as the NPV value for project X.
3. The above step is repeated for projects Y and Z.
4. Once all three NPV figures have been calculated, they are compared to find the highest NPV figure, and the program then displays the name of the project with the highest NPV.