Problem 1
Loan calculator

```csharp
private void loanPayment_Click(object sender, EventArgs e)
{
    double amount, annualRate, monthlyRate, payment;
    int years, months;

    // Get user input
    amount = Convert.ToDouble(loanPrincipal.Text);
    annualRate = Convert.ToDouble(loanRate.Text);
    years = Convert.ToInt32(loanTerm.Text);

    // Compute payment
    months = years * 12;
    monthlyRate = annualRate / 12;
    payment = monthlyRate * amount /
               (1 - (1 / Math.Pow(1 + monthlyRate, months)));

    // Display results
    results.Items.Add("The monthly payment is " + payment.ToString());
}
```

Problem 2
Credit card calculator

```csharp
private void minimumPayment_Click(object sender, EventArgs e)
{
    double currentBalance, monthlyRate, newBalance, minPayment;

    // Get user input
    currentBalance = Convert.ToDouble(cardBalance.Text);
    monthlyRate = Convert.ToDouble(cardRate.Text);

    // Compute this month's balance
    newBalance = currentBalance + currentBalance * monthlyRate;

    // Determine minimum payment
    if (newBalance < 10)
        minPayment = newBalance;
    else
    {
        minPayment = newBalance * .10;
        if (minPayment < 10)
            minPayment = 10;
    }

    // Display results
    results.Items.Add("The minimum payment is " + minPayment.ToString());
```
Problem 3
Approximation of e. This solution matches the example given with the last term added being the first one below the threshold.

```csharp
private void eCalculate_Click(object sender, EventArgs e)
{
    double sum, term, n, threshold;

    // Get user input
    threshold = Convert.ToDouble(eThreshold.Text);

    // Initialize everything for n = 1, so the sum starts out at 1
    sum = 1; // Includes 0th term
    n = 1;
    term = 1; // 0th term

    // Loop until term is under threshold
    while (term >= threshold)
    {
        // Compute next term
        term = term * 1 / n;

        // Add term to sum
        sum = sum + term;

        // Increment n
        n++;
    }

    // Display result
    results.Items.Add("Approximation of e: " + sum.ToString());
}
```

This solution is perhaps "better" in the sense that only terms above the threshold are added to the sum.

```csharp
private void eCalculate_Click(object sender, EventArgs e)
{
    double sum, term, n, threshold;

    // Get user input
    threshold = Convert.ToDouble(eThreshold.Text);

    // Initialize everything for n = 0, so the sum starts out at 0
    sum = 0;
    n = 0;
    term = 1; // 0th term

    // Loop until term is under threshold
    while (term >= threshold)
    {
        // Add term to sum
```
sum = sum + term;

// Increment n
n++;

// Compute next term
term = term * 1 / n;
}

// Display result
results.Items.Add("Approximation of e: " + sum.ToString());
}