Question 1: (1 point)

**Modify List Elements**

Create a list of size 75 and initialize all the values in that list to 0.

(a) Enter the list: __________

Create a new list that is the same as (a) except the value at position 9 is 1.

(b) Enter the new list: __________

(c) Finally, create a procedure called setList.
The procedure has two inputs a list L and a number 'a'. The procedure should make a copy of the list L and change the element at position 'a' to be the number 1. The procedure should return the new list you have created.

Enter only the procedure definition, the part beginning with **proc** and ending with **end proc**. Do not enter the trailing semi-colon after the **end proc**. 

setList := __________ ;

---

Question 2: (1 point)

**LeastSquares and Spline**
We want to look at the difference between LeastSquares and Spline fitted curves. A LeastSquares fitted curves creates a linear approximation of the data where the line tries to come close to as many of the points as possible. A spline curve approximates the data with a curve that goes through all the points.

We want to compare the results of different types of curve fitting. So we set up an experiment. The results we recorded are the following data:

time1 := [1, 2, 3, 4, 5, 6, 7, 8]
power := [60.89, 64.49, 70.39, 69.09, 73.99, 76.79, 75.09, 80.99]

When you plot the points (blue), LeastSquares (red) and Spline (green) curves together, you get the following plot:

Use a Degree 3 (cubic) Spline for the following questions.
(a) We want to evaluate each of these curves when t=4.53333
LeastSquares result: ____________  
Spline result: ____________  

(b) Again, we want to evaluate each of these curves when \( t = 6.22222 \)  
LeastSquares result: ______  
Spline result: ____________  

(c) The expressions we have created show power over time. The integral of either expression is the total amount of energy used.  
Integrate the Least Squares Line from 1 to 8:  
__________  
Integrate the Spline Curve from 1 to 8:  
__________  

---  

Question 3: (1 point)  

**Boolean Temperatures**  
In this question, you are going to create a Maple Procedure that returns true or false based on a given temperature. Creating functions like this is useful because the same code can be used to test for a variety of conditions by only changing the function. In part 3 of this question, use the procedure to filter a list of numbers to remove the items where your procedure returned false. Using a different procedure would allow you to filter a list for various reasons using the same code.  

1.) **Create a Procedure TempTest.**
The procedure should take one input which is a temperature. If the temperature is greater than or equal to 79 degrees then the procedure should return true. Otherwise, it should return false.

**Enter your procedure code.**
Enter only the procedure definition, the part beginning with `proc` and ending with `end` or `end proc`. Do not enter the trailing semi-colon after the `end`.

```
TempTest := __________;
```

2.) Generate a list of the true and false values for the following list of temperatures.

```
[89.08, 87.98, 73.24, 80.15, 60.05, 83.47, 81.47, 75.73, 82.44, 80.64, 87.91, 77.64, 88.10, 71.85, 72.97, 72.03, 63.11, 71.94, 76.18, 70.48, 86.49, 85.24, 84.26, 71.98, 80.55, 84.09, 71.97, 89.87, 84.40, 76.79, 68.96, 85.01, 70.62, 76.27, 86.30, 64.50, 82.86, 60.50, 75.88, 70.42, 83.93, 66.42, 68.16, 68.68, 73.80, 62.01, 74.97, 64.57, 61.54, 85.29, 86.31, 61.29, 73.64, 67.71, 61.71, 67.80, 81.94, 72.89, 81.90, 77.27, 65.20, 88.09, 67.79, 77.88, 63.07, 72.79, 64.08, 68.39, 73.84, 86.72, 71.13, 73.95, 61.82, 61.08, 60.16, 64.53, 63.07, 83.84, 79.53, 75.28, 72.36, 89.20, 77.31, 71.48, 84.94, 85.01, 76.99, 62.65, 62.95, 68.33, 69.38, 63.08, 75.96, 70.70, 62.83, 64.58, 65.14, 68.77, 61.06, 67.78, 71.88, 81.59, 73.54, 77.95, 81.97, 75.75, 83.61, 62.25, 80.58, 80.77, 86.93, 78.23, 73.19, 79.07, 89.67, 75.46, 66.80, 66.39, 70.81, 81.14, 89.36, 69.64, 86.89, 75.71, 67.00, 65.95, 62.23, 74.38, 67.98, 83.57, 62.40, 67.42, 83.69, 81.87, 77.12, 71.13, 67.82, 69.21, 71.23, 89.33, 79.27, 88.93, 76.86, 85.41, 86.32, 66.61, 85.44, 71.83, 72.49, 83.56]
```

Your answer should look like [true, false, false,...]

3.) Filter the list from question 2 to create a new list with only the temperatures your procedure returns true for.

Your answer should be a list of numbers. For example, [87.32,43.23,...]
Question 4: (2 points)

**Mortgage Interest**

A couple has bought a house for $302,441.00. They have taken out a mortgage for that amount from the bank at an annual interest rate of 3.71%.

To pay off the loan, the couple plan to make 12 payments of $1,986.00 spaced equally through the year. The bank does the following calculation with the payments:

It computes the interest owed for that period, which is the interest rate divided by the number of periods per year, times the balance still owed before the payment was processed. This is rounded to the nearest penny. The bank then subtracts the calculated interest for each paying period from the payment. What remains is deducted from the balance owed, reducing it a bit.

Use the built in maple round command for rounding to the nearest digit.

In the last period, the amount owed may be less than the regular payment. The bank asks for only the amount owed and then sets the balance owed to zero.

We are interested in calculating two things:

1. The number of periods it takes to pay off the loan, and
2. A list of numbers \( L \) corresponding to the value of the amount still owed at any particular period: \( L[1] \) is the original amount of the loan, \( L[2] \) is the amount still owed after one period, \( L[3] \), the amount owed after two periods, etc. The last element of \( L \) is always zero, assuming that the payments are sufficient to eventually pay off the loan. Each item in \( L \) should be a floating point number.

You can test your code with the following information:

\[
\begin{align*}
L[1] &= 302,441 \\
L[2] &= 301,390.05 \\
L[3] &= 300,335.85 \\
L[4] &= 299,278.39
\end{align*}
\]

(a) Compute and enter the complete list of floating point numbers \( L \) described
above:

(b) How many payments does it take to pay off the loan? Enter an integer (number without decimal point):

__________

Repeat the Process for a new situation.

A couple has bought a house for $260385.00. They have taken out a mortgage for that amount from the bank at an annual interest rate of 1.35%.

To pay off the loan, the couple plan to make 12 payments of $923.40 spaced equally through the year. The bank does the following calculation with the payments:
(c) Compute and enter the complete list of floating point numbers L:

__________

(c) How many payments does it take to pay off the loan? Enter an integer (number without decimal point):

__________

Question id: 35
Question 5: (2 points)

**Rectangle Integration**

As we have seen, Maple provides a exact method for calculating definite integrals. There are some cases where exact integration cannot be performed. Some programming languages do not support exact definite integration. In this case, the only way to integrate is to approximate. Another case is where we want to integrate something that is not defined by an exact symbolic expression. Rectangle Integration is a simple method to approximate an integral using loops. We create small rectangles using the function and sum the area to approximate the definite integral.

The above image shows a rectangle integration approximation of \( \int_{0}^{1} \sqrt{x} \, dx \) using 12 rectangles.

We will now examine the process more closely through an example.

We want to approximately integrate the \( \sqrt{x} \) function from 0 to 1. We will use
12 rectangles with equal widths in the x direction. This means the bottom of each rectangle as a size of 1/12.

We can call this value width and define it in general by stating:
width:=(Upper - Lower)/Number of rectangles in this case (1-0)/12.

We know the area of a rectangle is width*height. The height of a rectangle can be found by evaluating the function at specific value for x.

We now give the area for the first three rectangles:
Rectangle 0 Area = 1/12 * sqrt(0) = 0
Rectangle 1 Area = 1/12 * sqrt(1/12) = 0.02405626122
Rectangle 2 Area = 1/12 * sqrt(2/12) = 0.03402069088

For the ith rectangle the formula will be
Rectangle i Area = width * sqrt(Lower + i*width)

Summing all the areas gives an approximation for the integral.

The approximation of \( \int_{0}^{1} \sqrt{x} \, dx \) using 12 rectangles is 0.6202883616.

(a) Using Maple's Integration command find the exact value of \( \int_{0}^{1} \sqrt{x} \, dx \).

(b) Use the Rectangle Integration method described to approximate \( \int_{0}^{1} \sqrt{x} \, dx \) using 25 rectangles. Enter a floating point answer. It will be marked correct if it is within 2% of the correct answer.

(c) Use the Rectangle Integration method described to approximate \( \int_{0}^{1} \sqrt{x} \, dx \) using 1,569 rectangles. Enter a floating point answer. It will be marked correct if it is within 2% of the correct answer.

The below Maple procedure is based on a question from Quiz 3. Maple cannot find the definite integral of this procedure using the standard int() command. Our rectangle integration method will still be able to approximate this procedure's integral.

The procedure C is given below.
C:=proc(temp)
    local estimate,speed;
    if temp >= 78 then return 5654 end if;
    if temp <= 72 then return 2137 end if;
    estimate:=(3517/6 )*temp+(-40067);
    speed:=ceil(estimate/100)*100;
    return evalf(speed);
end proc;

(d) Use the Rectangle Integration method described to approximate \( \int_{72}^{78} C(x) \, dx \)

using 1,810 rectangles. Enter a floating point answer. It will be marked correct if it is within 2% of the correct answer.