1 Spherical Triangle

1.a Area and Excess

In this part all you have to do is to transfer the given mathematical formulas into assignments using arithmetic operators in MATLAB. When you run the script file, *test_triangle.m*, it will ask you to enter the values for three angles and the radius, then it will compute the area and excess using your formulas, and display the results.

```
function [Area, E] = spherical_triangle(A,B,C,R)
% Returns the area of a spherical triangle with
% spherical angles A,B,C

E    = A + B + C - pi;
Area = R^2 * E;
```

1.b Special Triangle

When the corner C is at North pole, and A & B are on the equator; the longtitudes of A and B, `LA` & `LB`, can be used to compute the value of the angle at C. The convention we followed was to consider the angle seen by the arc *starting from B, going in the west direction to reach A*. Depending on the values of `LA` and `LB` we have to branch our program.

```
LA = input('Enter the longtitude value for A in degrees: ');
LB = input('Enter the longtitude value for B in degrees: ');

if LA < LB
  C = (LB - LA)*pi/180;
else
  C = (360 + LB - LA)*pi/180;
end

A = pi/2; % longtitudes are perpendicular to the equator
B = pi/2;

R = input('Enter the value for the radius: ');

[Area, E] = spherical_triangle(A,B,C,R);

disp('The area of the special triangle is '); disp(Area);
disp('The excess value is '); disp(E);
```

1.c Testing Our Code

We can test our code using some special triangles we can recognize, like a hemisphere. You could have a script asking for user input or a fixture of test cases. The file `special_triangle.m` already contains the hemisphere as a test case in a separate code section. In order to make use of the function `spherical_triangle` you could write a script asking for values from the user:

```
A = input('Enter the angle  A: ');
B = input('Enter the angle  B: ');
C = input('Enter the angle  C: ');
R = input('Enter the radius R: ');

[Area, E] = spherical_triangle(A,B,C,R);

fprintf('The area of the triangle is %f\n', Area);
fprintf('The spherical excess is %f\n', E);
```

2 Distance

2.a Two Points

```
function dAB = distance(xA,xB,yA,yB,zA,zB)
% Returns the distance between points A and B which are specified
% using their cartesian coordinates in 3dimensions.

dAB = sqrt((xA-xB)^2+(yA-yB)^2+(zA-zB)^2);
```

2.b Triangular Area

```
function [Area,dAB,dBC,dCA] = triangle_area(xA,xB,xC,yA,yB,yC,zA,zB,zC)
% Returns the area and side lengths of a triangle. The input consists
% of the cartesian coordinates of vertices in 3-dimensions.

dAB = distance(xA,xB,yA,yB,zA,zB);
dBC = distance(xB,xC,yB,yC,zB,zC);
dCA = distance(xC,xA,yC,yA,zC,zA);

s = (dAB + dBC + dCA) / 2; % semi-perimeter

Area = sqrt(s*(s-dAB)*(s-dBC)*(s-dCA)); % dont forget *
```

2.c Testing Our Code

```
% Either use 'input' function, to get data from user, or use some test
% cases to check the your results.

% Case 1: Let's check for an equilateral triangle
xA = 1; yA = 0; zA = 0;
xB = 0; yB = 1; zB = 0;
xC = 0; yC = 0; zC = 1;
[A,a,b,c] = triangle_area(xA,xB,xC,yA,yB,yC,zA,zB,zC)
% We will later learn how to use 'assert' function to
% automatically check the test results.

% Case 2: Right-angled triangle with sides 3,4,5 on xy-plane
xA = 0; yA = 0; zA = 0;
xB = 4; yB = 0; zB = 0;
xC = 0; yC = 3; zC = 0;
[A,a,b,c] = triangle_area(xA,xB,xC,yA,yB,yC,zA,zB,zC)

% Case 3: ...
```

3 Distance Converter

Similar to the temperature converter which you worked on during one of the lab sessions:

```
function y = distance_converter()
  p = input('1) km to miles, 2) miles to km (enter 1 or 2) ');
  x = input('Enter the distance value: ');

  if p == 1
     y = x / 1.60934;
     fprintf('%f km is %f miles\n', x, y);
  elseif p == 2
     y = 1.60934 * x;
     fprintf('%f miles is %f km\n', x, y);
  else
     error('Wrong menu selection!');
  end
```

4 Setun Returns

One byte consists of 8 bits. We can represent $2^8 = 256$ values using a single byte. For the ternary system, we need to find the smallest integer x such that 3^x is just above 256, using MATLAB as a calculator: `ceil(log(256)/log(3))`, results in 6 trits.

5 Soccer Ball Toss

5.a Coin Toss

We can simulate a coin toss by generating a random number using `rand` function and then comparing it with 0.5. If *heads* is represented by the first half of the unit interval, the following function satisfy the requirements:

```
function x = coin_toss()
x = rand > 0.5;
```

5.b Ball Toss

The probabilities of *black* and *white* are proportional to the areas covered by *pentagons* and *hexagons*, respectively, on the spherical surface. The computation of the exact ratio is a little complicated, but you can use an approximation by considering a truncated icosahedron.

Let the side length of the hexagons and pentagons be a. Then the total area covered by the white regions can be computed as follows:

$$A_W = 20 \times \frac{3a^2\sqrt{3}}{2} \tag{1}$$

Similarly, the area covered by the pentagons is:

$$A_B = 12 \times \frac{a^2\sqrt{25+10\sqrt{5}}}{4} \tag{2}$$

We can compute the probabilities for black and white as:

$$P_B = \frac{A_B}{A_B + A_W} \approx 0.2843 \tag{3}$$

$$P_W = \frac{A_W}{A_B + A_W} \approx 0.7157 \tag{4}$$

We can modify the `coin_toss` function to mimic a ball toss as:

```
function x = ball_toss()
x = rand > 0.2843;
```

A more accurate result can be computed using the spherical triangle area equation and spherical trigonometric identities, which gives the critical value around: $P_B \approx 0.28167$, $P_W \approx 0.71833$.