Solution for Assignment 1

1. Give the IEEE single precision binary representation of each of the following decimal numbers:
   +2
   Answer. 0 10000000 00000000000000000000000
   −33
   Answer. 1 10000100 00001000000000000000000
   +1.3
   Answer. Answer: 0 01111111 01001100110011001100110
   −0.4
   Answer. 1 01111101 10011001100110011001101

2. How many IEEE single precision numbers x satisfy $1.0 \le x < 2.0$?
   Answer. The single precision numbers are 1.00000000000000000000000, 1.00000000000000000000001, 1.00000000000000000000010, $\cdots$, 1.11111111111111111111111. Thus there are 2^{23} numbers in the interval.

3. Consider the following program:

```
h = 1.0/2.0;
s = 2.0/3.0 - h;
t = 3.0/5.0 - h;
d = (s + s + s) - h;
n = (t + t + t + t + t) - h;
q = n/d;
```

   The variable q can take on different values depending on the floating-point system used by the computer.

   (a) Figure out the value of q, if the program is run in MATLAB/Octave (double precision). Explain the result.

   (b) Figure out the value of q, if the program is run in single precision. Explain your result.

   (c) Figure out the value of q, if the program is run on a hypothetical machine with $\beta = 10$, $t = 4$, $e_{\min} = -48$, and $e_{\max} = 49$.

   Answer.

   (a) The binary of h: $1.000...000 \times 2^{-1}$; The binary of s: $1.010...1010100 \times 2^{-3}$; The binary of t: $1.1001100...110011000 \times 2^{-4}$; The value of $(s+s+s)$ is: $1.111...110 \times 2^{-2}$, then the value of d is -2^{-53}; The value of $(t+t+t+t+t)$ is: $1.111...110 \times 2^{-2}$, then the value of n is -2^{-53}. Thus the value of q is 1.

   (b) The binary of h: $1.000...000 \times 2^{-1}$; The binary of s: $1.010...10101100 \times 2^{-3}$; The binary of t: $1.1001100...110011010000 \times 2^{-4}$; The value of $(s+s+s)$ is: $1.00...0001 \times 2^{-1}$, then the value of d is 2^{-24}; The value of $(t+t+t+t+t)$ is: $1.00...010 \times 2^{-1}$, then the value of n is 2^{-23}. Thus the value of q is 2.

(c) The decimal of h: 5.000×10^{-1}; The binary of s: 1.667×10^{-1}; The binary of t: 1.000×10^{-1}; The value of $(s+s+s)$ is: 5.001×10^{-1}, then the value of d is 10^{-4}; The value of $(t+t+t+t+t)$ is: 5.000×10^{-1}, then the value of n is 0. Thus the value of q is 0.

4. In 250 B.C.E. the Greek mathematician Archimedes estimated the number π as follows. He looked at a circle with diameter 1, and hence circumference π. Inside the circle he inscribed a square. The perimeter of the square is smaller than the circumference of the circle, and so it is a lower bound for π. Archimedes then considered an inscribed octagon, 16-gon, etc., each time doubling the number of sides of the inscribed polygon, and producing ever better estimates for π. Using 96-sided inscribed and circumscribed polygons, he was able to show that $223/71 < \pi < 22/7$. There is a recursive formula for these estimates. Let p_n be the perimeter of the inscribed polygon with 2^n sides. Then $p_2 = 2\sqrt{2}$. In general,

$$p_{n+1} = 2^n \sqrt{2(1 - \sqrt{1 - (p_n/2^n)^2})}$$

Compute p_n for $n = 3, 4, ..., 60$. Try to explain your results.
Kahan suggested a revision:

$$p_{n+1} = 2^n \sqrt{r_{n+1}}$$

where r_{n+1} can be computed iteratively

$$r_{n+1} = \frac{r_n}{2 + \sqrt{4 - r_n}} \qquad r_3 = \frac{2}{2 + \sqrt{2}}.$$

Use this revision to calculate r_n and p_n for $n = 3, 4, ..., 60$. Try to explain your results.
Answer. Two programs:

```
function p = pi1(niter)
% Usage: p = pi1(niter)
%
% Archimedes' method for computing pi
% Returns approximations to pi in vector p
%
% Input:
%   niter  number of interations

p(1) = 1.0;
p(2) = 2*sqrt(2);
power = 2.0;
for n=2:niter
    power = power*2;
    p(n+1) = power*sqrt(2*(1.0 - sqrt(1.0 - (p(n)/power)*(p(n)/power))));
end
```

```
function p = pi2(niter)
% Usage: p = pi2(niter)
%
```

```
% Kahan's revision of Archimedes' method for computing pi
% Returns approximations to pi in vector p
%
% Input:
%   niter  number of interations

p(1) = 1.0;
r(1) = 4.0;
power = 1.0;
for n=1:niter
    power = power*2;
    r(n+1) = r(n)/(2.0 + sqrt(4.0 - r(n)));
    p(n+1) = power*sqrt(r(n+1));
end
```

In the first program, p_{15} has 9 digits of accuracy, however, $p_{29} = 4.0$ and $p_{30} = 0.0$. The problem is the combination of rounding error and catastrophic cancellation. When $n = 28$, $(p_{28}/2^{28})^2 \approx 2^{-52}$, and $\sqrt{1-(p_{28}/2^{28})^2} \approx 1-(1/2)\times 2^{-52}$, which contains rounding error. Then catastrophic cancellation occurs in computing $1-\sqrt{1-(p_{28}/2^{28})^2} \approx 2^{-53}$. Thus $p_{29} \approx 2^{28}\sqrt{2\times 2^{-53}} = 4.0$. Once $p_{29} = 4.0$, $1-(p_{29}/2^{29})^2 = 1.0 - 2^{-54} = 1.0$ in double precision due to rounding error. Thus $p_{30} = p_{31} = \cdots = 0$. Note that p_{15} is the most accurate approximation since $(p_{14}/2^{14})^2 \approx \sqrt{u}$.
The Kahan's version eliminates cancellation.