Problems for Lecture 1

Name: Xin Chen
Student Number: A0151856Y

problem 1.

(a) Statement `char *s[4] = {"this", "that", "we", "!"};`
declares a pointer array each of which is pointing to the beginning of a string. In this case, `s` is also a pointer pointing to the beginning of array `s[4]`. Then `s[2][1]` means `*(*(s+2)+1)`, which gives the value `'e'`. similarly, `s[0][0]` is `'t'` and `s[0][4]` is `'\0'`.

(b) As said in (a), `s` is a pointer to pointer, so we can declare a function like `void myfunc(char **s)`, and use `myfunc(s)` to call the function.

If the array is declared as `char t[4][5]`, it can be passed to a function like `void myfunc(char t[4][5])` by calling `myfunc(s)`.

problem 2.

(a) **0.0:** `0 00000000 00000000000000000000000` (the exponent part is coded by eight zeros because of stipulation of IEEE)

1.0: `0 01111111 00000000000000000000000`

1/3: The science notation of 1/3 in binary system is:

$$0.01010\ldots$$

.
If we code this as

`0 01111111 01010101010101010101010`,
because now 0<e<255, the computer will retrieve the float by $(1+f)\times 2^{e-127}$. It's wrong because the "$1+f$".

So we need to rewrite 1/3 as

$$1.01010\ldots\times 2^{-2}$$

,
then the exponent is now $-2+127=125$.
So finally, the answer is
`0 01111101 01010101010101010101010`

(b)
machine epsilon 2^{-23}.

smallest positive number: $1 \times 2^{-23} \times 2^{-126} = 2^{-149}$ (form 1) or $(1+0) \times 2^{-126}$ (form 2).

Because $e = 255$ is for special cases, so the e for largest positive number is $e = 254$, then the real exponent is $254 - 127 = 127$.
So the largest positive number can be calculated by
$(1 + 2^{-1} + \cdots + 2^{-23}) \times 2^{127} = (2 - 2^{-23}) \times 2^{127}$.

problem 3.

Substitute $F_n = Ar^n$ to the recursion formula, then it's easy to find

$$r^2 + r - 1 = 0$$

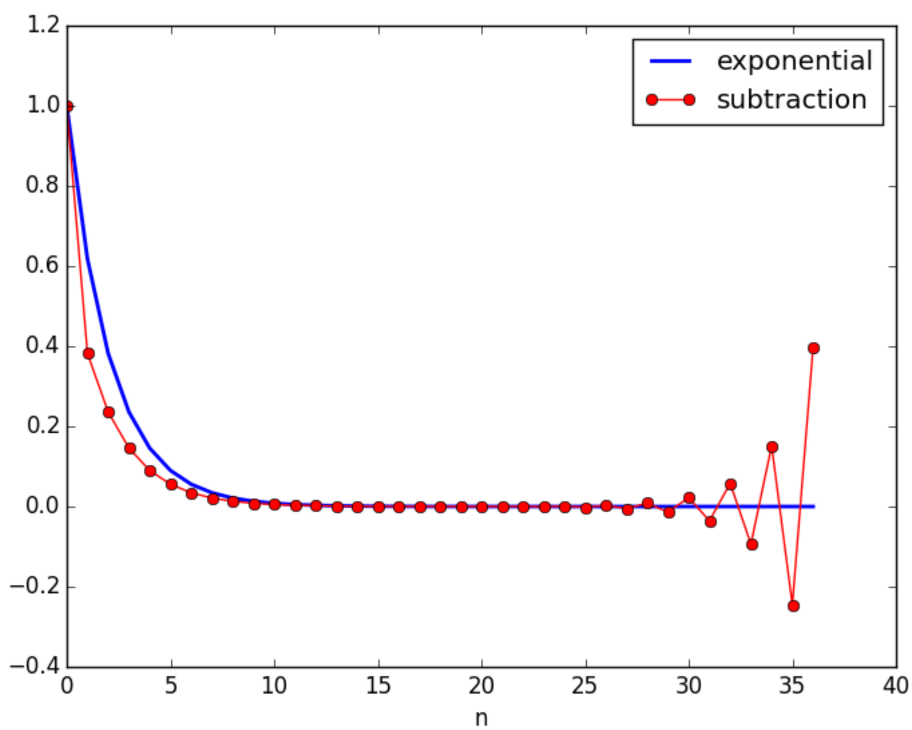
.

The solution is $r = \frac{-1 \pm \sqrt{5}}{2}$. So the general solution is

$$F_n = A\left(\frac{-1 \pm \sqrt{5}}{2}\right)^n$$

.

For the golden-mean case, $r = \frac{-1+\sqrt{5}}{2}$ and $A = 1$. Because $r < 1$, the Φ_n will go to 0 when increasing n and two adjacent number will be closer and closer. This just leads the instability of computation, which is clearly shown in the following figure.



(32-bit-float numbers are used in this plot.)