Homework #1

Due Time: 2016/10/20 (Thu.) 14:20
Contact TAs: `ada01@csie.ntu.edu.tw`

Instructions and Announcements

- There are two topic exercises and four regular problems. Topic exercises are easier than regular problems and are part of your mini-homeworks.
- **Programming.** The judge system is located at `https://ada01-judge.csie.org`. Please login and submit your code for the programming problems (i.e., those containing "Programming" in the problem title) by the deadline. NO LATE SUBMISSION IS ALLOWED.
- **Hand-written.** For other problems (also known as the "hand-written problems"), please submit your answers to the instructor at the beginning of the class. NO LATE SUBMISSION IS ALLOWED.
- **Collaboration policy.** Discussions with others are strongly encouraged. However, you should write down your solutions **in your own words**. In addition, for **each and every** problem you have to specify the references (e.g., the Internet URL you consulted with or the people you discussed with) on the first page of your solution to that problem. You may get zero point for problems due to the lack of references.
- Top-graded solutions/codes may be published as references for your fellow classmates.

Topic Exercise 1: Dynamic Programming (Programming)

Description

For a given integer sequence $a_1, a_2, \cdots, a_n$, we define $F(l, r) = \sum_{i=l}^{r} a_i$. Find the maximum value of $F(l, r)$ for all $1 \leq l \leq r \leq n$.

Input Format

The input contains two lines. The first line contains an integer n, the length of the integer sequence. The second line contains n integers, which are $a_1, a_2, \cdots, a_n$.

- $1 \leq n \leq 200000$
- $-10^5 \leq a_i \leq 10^5$

Output Format

Please output the maximum value in one line.

Sample Input	Sample Output
4 1 3 -2 100	102
5 -1 -3 -2 -4 -100	-1

Topic Exercise 2: Divide and Conquer (Programming)

Description

Here is an ancient puzzle.

There are three rods, and n disks are put at the first rod. Each disk has a different size. The disk size decreases from top to bottom, and they are numbered from 1 to n in this order.

The objective of the puzzle is to move the entire stack to another rod, obeying the following simple rules:

1. Only one disk can be moved at a time.
2. Each move consists of taking the uppermost disk from one of the stacks and placing it on top of another stack. That is, a disk can only be moved if it is the uppermost disk on a stack.
3. No disk may be placed on top of a smaller disk.

Now, you are given the size of n and the nicknames of each rod. All the disks will be put at the first rod initially and you need to move them to the third rod. Your program has to output a solution with the minimum steps.

Input Format

There are only four lines.

The first line contains an integer n indicating the number of disks. The second, third and fourth lines contain three lowercase strings s_1, s_2, s_3, respectively. They are the nicknames of rods.

- $1 \leq n \leq 16$
- $1 \leq |s_i| \leq 10$

Output Format

The first line is an integer m indicating the minimum steps m. Then, output m lines of your solutions. Please reference Sample Output for the output format.

Sample Input

```
3
first
second
third
```

Sample Output

```
7
1: move #1 disk from rod first to rod third
2: move #2 disk from rod first to rod second
3: move #1 disk from rod third to rod second
4: move #3 disk from rod first to rod third
5: move #1 disk from rod second to rod first
6: move #2 disk from rod second to rod third
7: move #1 disk from rod first to rod third
```

Problem 1

(1) Solve the recurrences. Assume that $T(1) = 1$.

(a) (4%) $T(n) = 25T(\frac{n}{5}) + 256n$
Use the substitution method to prove that $T(n) = O(n^2)$.

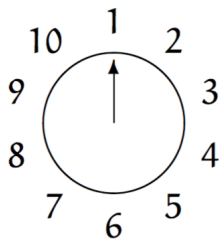
(b) (4%) $T(n) = 4T(\frac{n}{2}) + 10n^2$
Use a recursion tree to prove that $T(n) = O(n^2 \log n)$.

(2) (6%)Calculate the tight bound of the recurrence $T(n) = 4T(\frac{n}{2}) + n^2 \log n$. You are allowed to use any methods. Assume that $T(1) = 1$.

(3) (6%)Vanellope is a cute little girl who loves cookies. After her mother told her a Chinese folk tale, "the lazy wife", she decides to make a yummy cookies ring by herself and eat one cookie everyday. The cookies ring is made of n distinct cookies, labeled from 1 to n. To be creative, she decides to eat every second remaining cookie starting from 1 until no cookie is left.

Sadly, she finds out that she has only one chocolate cookie, which is her favorite. Therefore, she decides to leave the chocolate cookie to the last one she will eat.

For example, below is a starting configuration of a cookies ring of size 10:



The order to eat the cookies is 2,4,6,8,10,3,7,1,9,5, so 5 is the position where the chocolate cookie should be.

Let $S(n)$ be the position of the cookie last eaten when there are n cookies in total. Please help Vanellope form the recurrence relation of S(n), so she can calculate where to put the chocolate cookie. Please explain your answer.

Problem 2

Given two n-bit integers x and y, please follow the instructions bellow to design algorithms to solve the integer multiplication problem.

(1) (3%) Write pseudocode for solving integer multiplication by long multiplication, which multiplies the multiplicand by each digit of the multiplier and then add up all the properly shifted results. Then, give the time complexity of the algorithm using Θ notation with a brief explanation.

(2) (5%) Let $x = x_1 \cdot 2^{\frac{n}{2}} + x_0$, $y = y_1 \cdot 2^{\frac{n}{2}} + y_0$,

$$\begin{aligned} xy &= (x_1 \cdot 2^{\frac{n}{2}} + x_0)(y_1 \cdot 2^{\frac{n}{2}} + y_0) \\ &= x_1 y_1 \cdot 2^n + (x_1 y_0 + x_0 y_1) \cdot 2^{\frac{n}{2}} + x_0 y_0 \end{aligned}$$

Write pseudocode for solving integer multiplication by dividing the original n-bit problem (xy) into four $\frac{n}{2}$-bit subproblems (x_1y_1, x_1y_0, x_0y_1, and x_0y_0) recursively. Then, write down the running time $T(n)$ in a recurrence form and solve it by the master method to obtain the asymptotic "Θ" bound.

(3) (12%) In fact, we can improve the asymptotic time complexity of (2) by dividing the original n-bit problem (xy) into three $\frac{n}{2}$-bit subproblems recursively instead of four. Write pseudocode and running time $T(n)$ in a recurrence form for this algorithm. Then, solve $T(n)$ by the master method to obtain the asymptotic "Θ" bound.

Hint: You may find that the idea of Strassen's method inspiring.

Problem 3 - Red Persimmons (Programming)

Time limit : 1s

Description

(30%) In the supermarket there are N fruit boxes, each with K fruits inside. There are only two kinds of fruits: *Red Persimmons* and *Red Ceriums*. They basically looks the same, but *Red Ceriums* has more metal nutritions.

A box of fruit can be represented by a string of length K, where `P` means *Red Persimmons* and `C` means *Red Ceriums*. For example, "`PCCPC`" corresponds to a fruit box with 1st and 4th fruits being *Red Persimmons*, others being *Red Ceriums*.

HH randomly picks 2 boxes of fruit, and eats one from each box every day. He eats fruits in order, that is, he eats the i-th fruit of each box on the i-th day.

It's healthier to eat different fruits every day. So we can define the **Miserableness** to be the number of days HH needs to wait until he can eat different fruits, or just K if he can never eat different fruits. Formally, if the boxes he choose has string representation s_1 and s_2, then **Miserableness** is the length of *longest common prefix* of s_1, s_2, that is, the largest k such that $s_1[1..k] = s_2[1..k]$.

Here comes the problem. If HH choose boxes uniformly at random, what is the expected **Miserableness** he will experience?

Input Format

The first line contains two integers N, K. For the following N lines, the i-th line contains a string s_i with length K, where s_i is the string representation of the i-th fruit box.

- $N \geq 2$
- $K \geq 1$
- $N \times K \leq 1\,000\,000$
- s_i contains only `P` and `C`

- Test group 0 (0 points) : Sample testcases
- Test group 1 (20 points) : $N \times K \leq 2\,000$
- Test group 2 (40 points) : $N \leq 100\,000, K \leq 20$
- Test group 3 (40 points) : Original constraints

Output Format

Please output a real number, which is the expected Miserableness.

Your answer will be considered correct if the absolute or relative error is not greater than 10^{-6}. That is, if the correct answer is a, your answer is b, it must satisfy $|b-a| < 10^{-6}$ **or** $|b-a| < 10^{-6} \cdot a$.

Sample Input 1

```
2 5
PCCPC
PCPCP
```

Sample Output 1

```
2.0000000000
```

Sample Input 2

```
3 3
CCC
CCP
PCP
```

Sample Output 2

```
0.6666666667
```

Hint

- In the first sample, there is only one possible pair, and their longest common prefix is *PC*. So the expected Miserableness is 2.
- In the second sample, there are 3 possible pairs. If he chooses the first and second boxes, the Miserableness is 2; otherwise, it's 0. So the expected Miserableness is $\frac{2}{3}$.

Problem 4 - Pokemon Tower (Programming)

Time limit : 10s

Description

(30%) Again there's a new game about Pokemon: *Pokemon Tower*!

This game is very different from traditional series of Pokemon games. It consists of N levels, and you must go from level 1 through level N, and catch many Pokemons during that. After level N, there is a final boss battle.

Initially you have only one Pokemon: Pikachu! In each level there are several battles, and after winning all of them, you can choose a new Pokemon, or get \$100 as reward. Paticularly, for level k, there are a specific set S_k of Pokemons to choose, and you can take **at most one** from them. Also, it's impossible to obtain the same Pokemon in different levels, that is, if $i \neq j$, then $S_i \cap S_j = \emptyset$. Also it's impossible to get a second Pikachu.

There are many strategy for this game. To win the final battle, you can collect as many Pokemons, or only choose several Pokemons and use money to train them to legend levels. So we may wonder that: How many ways can you choose Pokemons throughout the game?

Let me solve this for you: if the number of candidate pokemons in level k is $p_k = |S_k|$, then the answer is just:

$$(p_1 + 1)(p_2 + 1) \cdots (p_N + 1)$$

So easy, right? Now there's a harder problem for you: How many ways can you choose Pokemons, if you want **exactly** K Pokemons in the end? Please calculate it for all K with $0 \leq K \leq N + 1$.

Input Format

The first line contains one integer N. The second line contains N integers, the i-th integer p_i is the number of choosable Pokemons in level i.

- $1 \leq N \leq 131\,072$
- $1 \leq p_i \leq 10^9$

- Test group 0 (0 points) : Sample testcases
- Test group 1 (20 points) : $N \leq 10, p_i \leq 2$
- Test group 2 (20 points) : $N \leq 2\,000$
- Test group 3 (30 points) : $p_i \leq 2$
- Test group 4 (30 points) : Original constraints

Output Format

Please output $N + 2$ space-seperated integers in a line. Those integers are $A_0, A_1, \cdots, A_{N+1}$ respectively, and A_K is the number of ways to get exactly K Pokemons in the end. As the answer can be very large, you should output the answer mod $1\,000\,000\,007$.

Sample Input 1

```
2
2 3
```

Sample Output 1

```
0 1 5 6
```

Sample Input 2

```
4
1 1 1 1
```

Sample Output 2

```
0 1 4 6 4 1
```

Sample Input 3

```
10
1 2 3 4 5 6 7 8 9 10
```

Sample Output 3

```
0 1 55 1320 18150 157773 902055 3416930 8409500 12753576 10628640 3628800
```

Sample Input 4

```
9
314 159 265 358 979 323 846 264 338
```

Sample Output 4

```
0 1 3846 6253512 666024911 437594513 124517822 64385960 584066593 90504701 846406502
```

Hint

- In the first sample: there's 1 way to choose nothing, 2 ways to choose only from level 1, 3 ways to choose only from level 2, and 6 ways to choose from both levels.
- Don't forget your Pikachu!