CPE preparation note

Vito's family

- 中位數 s[n/2] 就是Vito的家
- sort (s, s+r) 從 s 這個位址 排序到 s + r 這個位址

```
#include <bits/stdc++.h>
using namespace std;

int main ()
{
    int t, r;
    cin >> t;

    while ( t-- )
    {
        cin >> r;
        int s[r], sum = 0;
        for (int i = 0; i < r; i++)
        {
            cin >> s[i];
        }
        sort (s, s+r);

        int mid = s[r >> 1];
        for (int i = 0; i < r; i++)
        {
            sum += abs(s[i] - mid);
        }

        cout << sum << "\n";
    }

    return 0;
}
```

Hashmat the brave warrior

- 沒啥技巧 型別空間要夠大 大於2^32所以要用 64 bits 的 long long int

```cpp
#include <bits/stdc++.h>
using namespace std;

int main ()
{
    long long int a, b;
    while ( cin >> a >> b )
    {
        cout << abs(b - a) << "\n";
    }
    return 0;
}
```

You can say 11

- 套用檢驗11倍數的方法
- getline (cin, entry) 從 cin 流中讀一列字元直到碰到 '\n'（因為沒有指定delim）

```cpp
#include <bits/stdc++.h>
using namespace std;

int main()
{
    string entry;

    while ( getline( cin, entry ) && entry != "0")
    {
        bool isOdd = true;
        short sumOdd = 0;
        short sumEven = 0;
        for ( char c : entry )
        {
            switch (isOdd)
            {
                case 0:
                    sumEven += (atoi(&c));
                    break;
                case 1:
                    sumOdd += (atoi(&c));
                    break;
                default:
                    break;
            }
            isOdd = !(isOdd);
        }

        if( abs(sumOdd - sumEven) % 11 == 0 )
        {
            printf("%s is a multiple of 11.\n", entry.c_str());
        }
        else
        {
            printf("%s is not a multiple of 11.\n", entry.c_str());
        }
    }

    return 0;
}
```

Decode the Mad Man

- 技巧在於要將鍵盤上的字元順序先存下來（例如存成一個string）
- '\'前面要加一個跳多符號 '\'

- fgets (str, 1000, stdin) => 從 stdin 流中最多抓 1000 bytes 進去 str
- arr [len - 1] = '\0'

```
#include <bits/stdc++.h>
using namespace std;

int main ()
{
    string keyboard = "`1234567890-=qwertyuiop[]\\asdfghjkl;'zxcvbnm,./";

    char str[10000];
    char arr[10000];

    while (fgets(str, 1000, stdin))
    {
        int len = strlen(str);
        for(int i = 0; i < len - 1; ++i)
        {
            if(str[i] == ' ')
               arr[i] = ' ';
            else{
               str[i] = tolower(str[i]);
               char k = str[i];
               int pos = keyboard.find(k);
               arr[i] = keyboard[pos-2];
            }
        }
        arr[len-1] = '\0';
        printf("%s\n", arr);
    }
}
```

Primary Arithmetic

- scanf () != EOF

```cpp
#include<iostream>
#include<cstdio>
using namespace std;

int main()
{
    int add1, add2, carry, count;
    while( scanf( "%d%d", &add1, &add2 ) != EOF && (add1 || add2) )
    {
        carry = 0;
        count = 0;
    while( add1 || add2 )
    {
        carry = add1%10 + add2%10 + carry;
        carry /= 10;
        add1 /= 10;
        add2 /= 10;
        if( carry )
            count++;
    }
    if( count == 1 )
        printf( "1 carry operation.\n" );
    else if( count > 1 )
        printf( "%d carry operations.\n", count );
    else
        printf( "No carry operation.\n" );
    }
    return 0;
}
```

List of Conquests

- map <string, int> 因為沒有在意名字是什麼, 所以只統計人數

```cpp
#include <bits/stdc++.h>

using namespace std;

int main() {
    int case;
    cin>>case;
    map<string, int> ans;
    while (case--) {
        string s;
        cin>>s;
        ans[s]++;
        cin>>s>>s;
    }
    for (auto it = ans.begin(); it != ans.end(); ++it) {
        cout << it->first << " " << it->second << endl;
    }
    return 0;
}
```

What's Cryptanalysis?

- 利用 ASCII code 來代表每個字母的在int count[]內的索引

```cpp
#include <bits/stdc++.h>
using namespace std;

int main() {

    int count[256] = {0};
    int length = 0;
    char text;
    while (cin >> text)
    {
        length++;
        count[toupper(text)]++;
    }

    while(--length)
    {
        for(text = 'A'; text <= 'Z'; text++)
        {
            if(count[text] == length)
            {
                cout << text << " " << count[text] << endl;
            }
        }
    }
    return 0;
}
```

Summing digits

- 利用 string 來存輸入的數字, 在一個位數一個位數轉

```cpp
#include <iostream>
#include <string>
#include <numeric>
using namespace std;

int main(){
    string tmp;
    int sum;

    while(cin >> tmp){
        if (tmp == "0"){
            break;
        }
        if (tmp.length() == 1){
            cout << tmp << endl;
            continue;
        }

        while (tmp.length() > 1){
            sum = 0;
            int num[10] = {0};
            for (int i = 0; i < tmp.length(); i++){
                num[i] = (int)tmp[i] - '0';
                sum += num[i];
            }
            tmp = to_string(sum);
        }
        cout << sum << endl;
    }

    return 0;
}
```

Odd sum

```
#include<stdlib.h>

int main(){

    #include <iostream>
using namespace std;

int main(){
    int t, a, b, sum;

    cin >> t; // size of test case
    for (int i = 0; i < t; i++){
        cin >> a >> b;

        sum = 0;

        if (a % 2 == 0){
            a++;
        }

        while (a <= b){
            sum = sum + a;
            a += 2;
        }
        cout << "Case " << i+1 << ": " << sum << endl;
    }


    return 0;
}
}
```

Common permutation

```cpp
#include <iostream>
#include <string>
#include <algorithm>
using namespace std;

int main(){
    string a, b;
    int countA, countB;

    char tmp;

    while (getline(cin, a) && getline(cin, b)){
        countA = 0;
        countB = 0;
        int letter[26] = {0};

        for (int i = 0; i < 26; i++){
            countA = count(a.begin(), a.end(), 97+i);
            countB = count(b.begin(), b.end(), 97+i);

            letter[i] = min(countA, countB);

            if (letter[i] != 0){
                for (int j = 0; j < letter[i]; j++){
                    tmp = 97 + i;
                    cout << tmp;
                }
            }
        }
        cout << endl;
    }
    return 0;
}
```