Jan 18

UVa 11321 — Sort! Sort!! and Sort!!!

題目連結：UVa 11321
難度：一顆星

題目大意：
測資會給予數列有幾個數字及餘數值，接著為數列之數字，需依照題目之規則做升序排序。
規則如下，
(1) mod值較小者優先
(2) mod相等時，奇數優先
(3) 兩數皆為奇數時，數值大者優先
(4) 兩數皆為偶數時，數值小者優先

解題過程：
1. 定義 class 來放置要排序的數列之數值及其餘數
2. 運用 **sort** 函式自定義 compare function
（注意！不能直接用 x % 2 == 0 來判斷奇偶，當遇到負數時會出錯）

使用語言：C++

參考程式如下：

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

class Number{
    public:
        int number;
        int mod;
};

bool sortFunc(Number, Number);

int main(){
    int num, mod_value;

    while (cin >> num >> mod_value){
        if (num == 0 && mod_value == 0){
            break;
        }

        Number set[num];
        for (int i = 0; i< num; i++){
            cin >> set[i].number;
            set[i].mod = set[i].number % mod_value;
        }

        sort(set, set+num, sortFunc);

        cout << num << " " << mod_value << endl;
        for (int i = 0; i < num; i++){
            cout << set[i].number << endl;
        }
    }

    cout << "0 0" << endl;

    return 0;
}

bool sortFunc(Number x, Number y){
    if (x.mod == y.mod){
        if (abs(x.number) % 2 == 0 && abs(y.number) % 2 == 0){ // both are even
            return (x.number < y.number); // smaller even number < larger even number
        } else if (abs(x.number) % 2 == 1 && abs(y.number) % 2 == 1){ // both are odd
            return (x.number > y.number); // larger odd number < smaller odd number
        } else {
            if (abs(x.number) % 2 == 1){ // odd number < even number
                return true;
            }
            return false;
        }
    } else {
        return (x.mod < y.mod);
    }
}
```

Jan 18

UVa 11150 — Cola

題目連結：UVa 11150
難度：一顆星

題目大意：
測資會給予多組可樂瓶數，每三個空瓶可兌換一瓶可樂，計算最多可享用多少可樂。
（可與朋友借空瓶來兌換可樂，但最後要能夠歸還空瓶）

解題過程：
1. 用迴圈計算分別借 0~2 個空瓶時能享用的可樂數
2. 檢查最後是否能歸還空瓶，若否，則此種兌換方法不成立

使用語言：C++

參考程式如下：

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

int main(){
    int cola_num;

    while (cin >> cola_num){
        int max_cola = 0;

        for (int i = 0; i < 3; i++){ // number of empty bottle I borrow
            int drink_cola_sum = cola_num;
            int tmp_cola = 0;
            int tmp_empty = cola_num + i;

            while (tmp_empty >= 3){
                drink_cola_sum += tmp_empty / 3;
                tmp_empty = tmp_empty % 3 + tmp_empty / 3;
            }

            // can't return empty bottle to friend => can't borrow i bottles
            if (tmp_empty < i){
                drink_cola_sum = 0;
            }

            if (max_cola < drink_cola_sum){
                max_cola = drink_cola_sum;
            }
        }
        cout << max_cola << endl;
    }

    return 0;
}
```

Jan 18

UVa 118 — Mutant Flatworld Explorers

題目連結：UVa 118
難度：一顆星

題目大意：
測資會給予整體座標大小以及好幾個機器人的初始座標、方位，並要預測在一串指令後，機器人的所處位置。若過程中有機器人掉出邊界，會紀錄該位置（不管方位），避免有其他機器人也從同樣座標墜落。

解題過程：
1. 宣告一個 **int[][]** 來放置機器人掉落的座標位置（注意！須大於測資所給之整體座標大小，可參考第 *34* 行 ）
2. 宣告一個 **char[]** 來放置機器人朝向的方位，即可利用**加減索引**來表示機器人目前之面向方位
3. 用迴圈處理機器人之指令，並在前進指令後，檢查是否掉出邊界。若掉出邊界，先檢查是否為之前有機器人掉落之位置，若是，則**忽略**此次指令；反之，則判斷掉出邊界，並記錄此座標於 1. 之陣列中

使用語言：C++

參考程式如下：

```
1   #include <iostream>
2   #include <cstring>
3   #include <algorithm>
4   using namespace std;
5
6   class Robot{
7       public:
8           int x;
9           int y;
10          int lost; // yes: 1, no: 0
```

```cpp
        int ori_index;
        string instr;

        int is_lost(int x_bound, int y_bound){
            if (x > x_bound || x < 0){
                return 1;
            } else if (y > y_bound || y < 0){
                return 1;
            } else {
                return 0;
            }
        }
};

int main(){
    int upper_right_x, upper_right_y;
    Robot robot;
    char robot_ori[4] = {'E', 'N', 'W', 'S'};
    char origin_ori;

    cin >> upper_right_x >> upper_right_y;

    // initial map[][]
    int map[upper_right_x+1][upper_right_y+1];
    for (int i = 0; i <= upper_right_x; i++){
        for (int j = 0; j <= upper_right_y; j++){
            map[i][j] = 0;
        }
    }

    while (cin >> robot.x >> robot.y >> origin_ori){
        cin >> robot.instr;

        // find the index of origin_ori in robot_ori[]
        robot.ori_index = distance(robot_ori, strchr(robot_ori, origin_ori));
        robot.lost = 0;

        int tmp_x = robot.x;
        int tmp_y = robot.y;

        for (int i = 0; i < robot.instr.length(); i++){
            if (robot.instr[i] == 'L'){ // turn left
                robot.ori_index ++;
                if (robot.ori_index > 3){
                    robot.ori_index = 0;
                }
            } else if (robot.instr[i] == 'R'){ // turn right
                robot.ori_index --;
                if (robot.ori_index < 0){
                    robot.ori_index = 3;
                }
            } else if (robot.instr[i] == 'F'){ // move forward
                tmp_x = robot.x;
                tmp_y = robot.y;

                if (robot_ori[robot.ori_index] == 'E'){
                    robot.x ++;
                } else if (robot_ori[robot.ori_index] == 'N'){
                    robot.y ++;
                } else if (robot_ori[robot.ori_index] == 'W'){
                    robot.x --;
                } else if (robot_ori[robot.ori_index] == 'S'){
                    robot.y --;
                }

                // check if robot losts
                robot.lost = robot.is_lost(upper_right_x, upper_right_y);
                if (robot.lost){
                    robot.x = tmp_x;
                    robot.y = tmp_y;

                    if (map[robot.x][robot.y] == 1){
                        // It's the scent, won't move off the edge
                        robot.lost = 0;
                    } else {
                        map[robot.x][robot.y] = 1;
                        break;
                    }
                }
            }
        }

        // output
        cout << robot.x << " " << robot.y << " " << robot_ori[robot.ori_index];
        if (robot.lost == 1){
            cout << " LOST";
        }
        cout << endl;
    }
    return 0;
}
```

Jan 14

UVa 10415 — Eb Alto Saxophone Player

題目連結：UVa 10415
難度：一顆星

題目大意：
測資會給予一串音符，需計算每一指按按鈕的次數。
（若前一個音符有使用那指，則在目前的音符不需再計算一次）

解題過程：
1. 宣告一個二維陣列，紀錄每一個音符需按的指法
2. 用兩個 array 分別紀錄總按鈕次數及前一個指法，以進行比對
3. 注意！測資之歌曲可能為空行，因此使用 **getline()** 讀入測資

使用語言：C++

參考程式如下：

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

char notes[14] = {'c', 'd', 'e', 'f', 'g', 'a', 'b', 'C', 'D', 'E', 'F', 'G', 'A', 'B'};
int finger[14][10] = {
    {0,1,1,1,0,0,1,1,1,1}, // c
    {0,1,1,1,0,0,1,1,1,0}, // d
    {0,1,1,1,0,0,1,1,0,0}, // e
    {0,1,1,1,0,0,1,0,0,0}, // f
    {0,1,1,1,0,0,0,0,0,0}, // g
    {0,1,1,0,0,0,0,0,0,0}, // a
    {0,1,0,0,0,0,0,0,0,0}, // b
    {0,0,1,0,0,0,0,0,0,0}, // C
    {1,1,1,1,0,0,1,1,1,0}, // D
    {1,1,1,1,0,0,1,1,0,0}, // E
    {1,1,1,1,0,0,1,0,0,0}, // F
    {1,1,1,1,0,0,0,0,0,0}, // G
    {1,1,1,0,0,0,0,0,0,0}, // A
    {1,1,0,0,0,0,0,0,0,0}, // B
};

int main(){
    int case_num;
    string song;

    cin >> case_num;
    getline(cin, song); //clear buffer
    for (int i = 0; i < case_num; i++){
        string song;
        int f_num[10] = {0};
        int last_f[10] = {0};
        int note_index;

        getline(cin, song);
        for (int j = 0; j < song.length(); j++){
            // find the index of note to find the fingering
            note_index = distance(notes, strchr(notes, song[j]));

            if (j == 0){
                for (int k = 0; k < 10; k++){
                    f_num[k] = finger[note_index][k];
                    last_f[k] = finger[note_index][k];
                }
            } else {
                for (int k = 0; k < 10; k++){
                    // check if finger used in last note
                    if (finger[note_index][k] == 1 && finger[note_index][k] != last_f[k]){
                        f_num[k] ++;
                    }
                    last_f[k] = finger[note_index][k];
                }
            }
        }

        // output
        for (int k = 0; k < 10; k++){
            if (k != 0){
                cout << ' ';
            }
            cout << f_num[k];
        }
        cout << endl;
    }
    return 0;
}
```

Jan 14

UVa 10409 — Die Game

題目連結：UVa 10409

難度：一顆星

題目大意：
一開始骰子的正面為 1、北面為 2、西面為 3，依照測資所給之接續動作將骰子翻面，預測最後骰子之正面為何。

解題過程：
1. 題目有給提示：骰子上任何相對的面之數字總和為 7
2. 用 array 來記錄正面、北面、西面之點數
3. 透過 string.compare 來判斷骰子翻轉的方位，並找出往各方位轉動時，點數變化的規律，來做最後正面點數之預測

使用語言：C++

參考程式如下：

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

int main(){
    int command_num, tmp;
    string command;

    while(cin >> command_num){
        // end of input
        if (command_num == 0){
            break;
        }

        int dice[3] = {1, 2, 3}; // top, north, and west faces

        for (int i = 0; i < command_num; i++){
            cin >> command;

            tmp = dice[0];
            if (command.compare("north") == 0){
                dice[0] = 7 - dice[1];
                dice[1] = tmp;
            } else if (command.compare("south") == 0){
                dice[0] = dice[1];
                dice[1] = 7 - tmp;
            } else if (command.compare("west") == 0){
                dice[0] = 7 - dice[2];
                dice[2] = tmp;
            } else if (command.compare("east") == 0){
                dice[0] = dice[2];
                dice[2] = 7 - tmp;
            }
        }

        cout << dice[0] << endl;
    }

    return 0;
}
```

UVa10409.cpp hosted with ♡ by GitHub view raw

Jan 13

UVa 10189 — Minesweeper

題目連結：UVa 10189
難度：一顆星

題目大意：
用測資所給之地雷分佈，計算每個點的九宮格內有幾個地雷數。

解題過程：
1. 宣告兩個 array，一個放測資，一個放地雷數
2. 用迴圈一一尋找各個點周圍的地雷數
3. 注意在尋找地雷時，pointer 不要 <0 或 ≥ col/row
4. 注意輸出時的換行，在最後一組 Field 輸出後會有一個空行

使用語言：C++

參考程式如下：

```
#include <iostream>
#include <string>
using namespace std;
int find_mine(string[], int, int, int, int);

int main(){
    int col, row;
    int index = 1;

    while (cin >> col >> row){
        // end of input
```

```cpp
        if (col == 0 && row == 0){
            break;
        }

        string game[col];
        char mine_num[col][row];

        for (int i = 0; i < col; i++){
            cin >> game[i];
        }

        // find the number of adjacent mines
        for (int i = 0; i < col; i++){
            for (int j = 0; j < row; j++){
                if (game[i][j] != '*'){
                    mine_num[i][j] = '0' + find_mine(game, col, row, i, j);
                } else {
                    mine_num[i][j] = '*';
                }
            }
        }

        // output
        if (index != 1){
            cout << endl;
        }
        cout << "Field #" << index << ":" << endl;
        for (int i = 0; i < col; i++){
            for (int j = 0; j < row; j++){
                cout << mine_num[i][j];
            }
            cout << endl;
        }
        index++;
    }
    return 0;
}

int find_mine(string game[], int col, int row, int col_index, int row_index){
    int count_mine = 0;
    for (int x = col_index-1; x <= col_index+1; x++){
        if (x >= 0 && x < col){
            for (int y = row_index-1; y <= row_index+1; y++){
                if (y >= 0 && y < row){
                    if (game[x][y] == '*'){
                        count_mine++;
                    }
                }
            }
        }
    }
```

```
        return count_mine;
}
```
UVa10189.cpp hosted with ♡ by GitHub view raw

Sep 3, 2020

2020 CSCS 實習心得

會申請 CSCS 實習，是因為我一直想要學習資訊安全卻找不到較有系統的學習方法，因此當初看到 2020 NGO 資安提升計畫實習生招募的資訊時，我便覺得這是一個很好的機會，也很幸運地獲得這次的實習機會。

實習期間所接觸的工作以及收穫

- 認識開放文化基金會(OCF)及開源
  在實習前我對開源概念的了解，大概只有 Github 而已xD，在和 OCF 的夥伴一起工作過後，我才了解開源的概念不只能用在軟體，開源能讓科技的發展變得更自由、更創新。也在這段期間接觸到很多陌生領域的知識，像是 g0v、RightsCon、COSCUP 等，感覺又增加了很多領域能夠探索！

有興趣的人，我推薦訂閱高資訊含量的 OCF Lab

- 參與 CSCS 社群活動
  實習期間是我第一次積極參與社群活動，過程中我有機會近距離了解社群的經營，也實際參與了社群的討論會議，甚至成為社群的一員。CSCS 社群對新人很友善，大家也都很樂於分享自己的經驗，交流各種資安新聞及資訊。過去我一直找不到學習資安的方法，但在跟一群優秀的人一起工作過後，也知道自己以後可以透過自己架設 VPN、在沙盒內嘗試惡意軟體等方法來學習更多資安知識。

- 校對 SEC 中譯教材
  SEC 教材是電子前哨基金會提供給資安教育者的教材，因此內容非常淺顯易懂，也非常豐富。在校對中文譯版的過程，我學習到如何轉譯知識，更在其中學習到大量的資安基礎觀念，讓我有能力去學習更深入的資安領域、資安防護，也學習到能如何運用於資安推廣。

SEC 教材中文版譯本 是很適合資安新手學習資安基礎知識的教材，推推！

- 協助製作 CSCS 教材模組
  CSCS 教材模組是為了提供 CSCS 社群的講師們一致性的參考教材而製作，剛開始看教材模組時，因為背景知識不足，我沒辦法提出實用的建議QQ；但是在校對 SEC 中譯教材及參與試教活動後，我漸漸能夠為教材增加一些新的案例、協助撰寫講師稿等調整可以變得更好的地方，最後也和 Lulu、Claire 一起修正出了更完整的第三版教材。

- 體驗資安健檢
  資安健檢是我最喜歡的部分，CSCS 講師會協助健檢者檢查電腦手機的設定以及辦公室的硬體設備有沒有資訊安全上的風險及漏洞。我所參與的這場資安健檢，讓我學到很多可以運用在日常卻容易忽略的小知識，從一開始講師 YuTing 的講解，到後來實際操作、協助其他人調整電腦及手機的設定來做好資安防護的過程，除了讓我學習到該怎麼在電腦及手機上調整應用程式的權限、使用哪些應用程式會產生風險、哪些工具可以協助檢測電腦是否有惡意軟體等知識，也覺得自己開始有能力在資安方面幫助其他人。

- 籌備東亞民主論壇 —— 資安培力工作坊
  從無到有，籌備一個工作坊是一件很有成就感的事情；資安培力工作坊是為了提供 NGO 工作者基礎的資安知識，使他們能夠具備挑選較安全的數位產品的能力，也使他們在工作及日常中能做好基礎的資安防護來保護自己的資安及隱私。
  在籌備工作坊的過程中，我能夠提出任何建議來讓工作坊變得更好；在大家的討論修改下，也讓工作坊從試教活動到工作坊當天有很大很不錯的改變。
  而試教活動及工作坊當天，我親身體驗多個講師的教學風格以及看見學員認真參與、提出各種問題的反應，讓我覺得也許在籌備工作坊的過程中，我才是收穫最多的人。

總結

在實習的兩個月中，我接觸了許多工作內容，也在這樣的過程裡，培養了自己的軟實力、硬實力。在 OCF 及 CSCS 團隊中，不管遇到什麼不懂的東西而提問，都會獲得非常詳盡而耐心的解答；在參與討論中也是真的能夠提出意見，跟著社群一起進步成長的。

在兩個月之中，我接觸到了很多之前不曾碰觸到的領域 —— 開源、NGO及各種大型年會，我才發現自己還有非常多能夠學習的管道，也還有很多想要關心的議題，我覺得自己非常幸運，能夠擁有這次機會，參與實習。

在實習之後，我開始進入資安的世界，從最初的知識學習，到實際操作、參與工作坊、編撰講義教材等等，讓我從一個僅僅對資安擁有興趣的學生，變成能具備良好實務經驗、能夠向大家推廣、分享的人，在未來，我也會為了資訊安全及社會議題不斷努力、繼續學習。

Jul 8, 2020

UVa 10226 — Hardwood Species

題目連結：UVa 10226
難度：一顆星

題目大意：
將測資給予之品種分類，計算其所佔比例。

解題過程：
1. 將品種設為物件，屬性：品種名稱、品種數量
2. 將測資讀入 (注意讀入形式)，在 **vector<品種>** record中尋找是否已存在；若存在則將其數量+1，反之則將其記錄於 record 內
3. 依 record.name 作排序 (由 a~z)
4. 輸出 record.name 及 其數量/樹之總數量

使用語言：C++

參考程式如下：

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

class species{
    public:
        string name;
        double num;
};

bool comp(species i, species j){
    return i.name < j.name;
}

int main(){
    int testCase_num;
    int treeNum;
    int isExist;
    vector<species> record;
    string tmpTree;

    cin >> testCase_num;
    for (int i = 0; i < testCase_num; i++){
        if (i != 0){
            record.clear();
            cout << endl;
        } else {
            cin.ignore();
            getline(cin, tmpTree); // ignore the line break after testCase_num
        }
        treeNum = 0;

        while (getline(cin, tmpTree)){
            if (tmpTree == ""){
                break;
            }
            // check if tmpTree exists in record[] and count its num
            isExist = 0;
            for (int j = 0; j < record.size(); j++){
                if (tmpTree == record[j].name){
                    isExist = 1;
                    record[j].num++;
                    break;
                }
            }

            // if tmpTree not exist in record[]
            if (!isExist){
                species tmpSpecies;
                tmpSpecies.name = tmpTree;
                tmpSpecies.num = 1;
                record.push_back(tmpSpecies);
            }
            treeNum++;
        }

        // in alphabetical order
        sort(record.begin(), record.end(), comp);

        // count the percentage of the each trees' population
        for (int j = 0; j < record.size(); j++){
            double num = record[j].num / treeNum * 100;
            cout.precision(4);
            cout << record[j].name << " " << fixed << num << endl;
        }
    }
    return 0;
}
```

UVa10226.cpp hosted with ♡ by GitHub   view raw

Jul 8, 2020

UVa 299 — Train Swapping

題目連結：UVa 299
難度：一顆星

題目大意：
將每筆測資之數列由小到大排序，並計算交換次數。
（相當於 **Bubble Sort**，一次比較兩兩相鄰元素，若前者較大則交換，一輪結束後，最大元素會落在陣列尾端）

解題過程：
1. 用迴圈比較兩兩相鄰元素，若前者較大則交換，並將交換次數++
2. 最後輸出總交換次數

使用語言：C++

參考程式如下：

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

int main(){
    int testCase_num, trainLength;
    int tmpTime, swapTimes;

    cin >> testCase_num;
    for (int i = 0; i < testCase_num; i++){
        swapTimes = 0;
        tmpTime = -1;

        cin >> trainLength;
        int carriage[trainLength];
        for (int j = 0; j < trainLength; j++){
            cin >> carriage[j];
        }

        // bubblesort
        while(tmpTime != 0){
            tmpTime = 0;
            for (int k = 1; k < trainLength; k++){
                if (carriage[k-1] > carriage[k]){
                    int tmp = carriage[k-1];
                    carriage[k-1] = carriage[k];
                    carriage[k] = tmp;

                    tmpTime++;
                }
            }
            swapTimes += tmpTime;
            trainLength--;
        }
        cout << "Optimal train swapping takes " << swapTimes << " swaps." << endl;
    }
    return 0;
}
```

UVa299.cpp hosted with ♡ by GitHub   view raw

Jul 4, 2020

UVa 10062 — Tell me the frequencies!

題目連結：UVa 10062
難度：一顆星

題目大意：
題目給定字串後，找出各個字元之 ASCII，並按照頻率由低到高印出。
(若頻率相同，則以 ASCII 較大者先印出)

解題過程：
1. 因為會有字串中會有空白，因此用 **getline(cin, str)** 讀入測資
2. 後續計算中若頻率相同，會以 ASCII 較大者先印出，且搜尋是從陣列[0] 開始，因此陣列中所放資料從 ASCII 126~32，來減少 6. 迴圈執行次數
3. 將計算頻率的陣列之初始值設為 >1000的值
(因頻率要從低到高排序，且字串長度不超過1000)
4. 用迴圈計算每個字元出現之頻率，並用 **(*int*)str[i]** 轉換為 ASCII
5. 用 **min_element(freq, freq+freq.length)**找出陣列中頻率最低的 address，並用 **distance(freq, address)** 求出其在陣列中的 index
6. 不斷迴圈進行 5. 直到全數出現過的字元皆印出

使用語言：C++

参考程式如下：

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

int main(){
    string str;
    int count = 0;
    while (getline(cin, str)){
        if (count > 0){
            cout << endl;
        }
        int freq[95]; // ascii 126~32
        for (int i = 0; i < 95; i++){
            freq[i] = 10000;
        }

        // calculate the frequency of each char
        int tmp = 0;
        for (int i = 0; i < str.length(); i++){
            tmp = 126 - (int)str[i];
            if (freq[tmp] == 10000){
                freq[tmp] = 0;
            }
            freq[tmp]++;
        }

        // find the least frequency char and print it
        int index;
        index = distance(freq, min_element(freq, freq + 95));
        tmp = freq[index];
        while (tmp != 10000){
            for (int i = index; i < 95; i++){
                if (freq[i] == tmp){
                    cout << 126 - i << " " << freq[i] << endl;
                    freq[i] = 10000;
                }
            }
            index = distance(freq, min_element(freq, freq + 95));
            tmp = freq[index];
        }
        count++;
    }
    return 0;
}
```

UVa10062.cpp hosted with ♡ by GitHub view raw

Jul 4, 2020

UVa 10057 — A mid-summer night’s dream.

題目連結：UVa 10057
難度：一顆星

題目大意：
求測資數列之最小中位數、中位數個數以及有多少整數符合中位數。

解題過程：
1. 將輸入的測資數列 num[] 進行排序，其 length = n
2. 取 num[n/2–1] 及 num[n/2]為中位數（偶數數列會有兩個，前者為min）
3. 用迴圈找數列中，符合上述兩個中位數的個數
4. 計算兩個中位數間存在多少個整數（其皆可能為此數列之中位數）

使用語言：C++

參考程式如下：

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

int main(){
    int n;

    while (cin >> n){
        vector<int> num;

        int tmp;
        for (int i = 0; i < n; i++){
            cin >> tmp;
            num.insert(num.end(), tmp);
        }

        sort(num.begin(), num.end());

        // find the minimum median in the sequence
        int minMedian = 0;
        int maxMedian = 0;
        if (n % 2 == 0){ // if n is even, the median of num is num[n/2] and num[(n/2)-1]
            minMedian = num[(n/2)-1];
            maxMedian = num[n/2];
        } else { // if n is odd, the median of num is num[n/2]
            minMedian = num[n/2];
            maxMedian = num[n/2];
        }

        // find the amount of number equal to the median in the sequence
        int count = 0;
        for (int i = 0; i < n; i++){
            if (num[i] == minMedian || num[i] == maxMedian){
                count++;
            }
        }

        // find the amount of possible different integer can be the median of sequence
        int possibleMedian = 0;
        if (n % 2 == 0){
            possibleMedian = maxMedian - minMedian + 1; // the amount between two medians
        } else {
            possibleMedian = 1;
        }

        cout << minMedian << " " << count << " " << possibleMedian << endl;
    }
    return 0;
}
```

UVa10057.cpp hosted with ♡ by GitHub view raw

Jun 21, 2020

UVa 10242 — Fourth Point !!

題目連結：UVa 10242
難度：一顆星

題目大意：
給定平行四邊形兩個相鄰邊端點的 (x, y) 坐標，求第四個點之坐標。

解題過程：
1. 判斷四個點中何者是重疊的，設為 second

2. 用向量計算 second->first 跟 second->third，並將兩個向量相加，作為始於 second 到 fourth 的向量

使用語言：C++

參考程式如下：

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

int main(){
    double pointX[4], pointY[4];
    int index[2]; // the index of overlapping point
    double firstX, firstY, secondX, secondY, thirdX, thirdY;
    double fourthX, fourthY;
    double vector1_X, vector1_Y, vector2_X, vector2_Y;

    while (cin >> pointX[0] >> pointY[0]){
        for (int i = 1; i < 4; i++){
            cin >> pointX[i] >> pointY[i];
        }

        // find the overlappig point
        for (int i = 0; i < 4; i++){
            for (int j = i+1; j < 4; j++){
                if (pointX[i] == pointX[j] && pointY[i] == pointY[j]){
                    index[0] = i;
                    index[1] = j;
                }
            }
        }

        secondX = pointX[index[0]];
        secondY = pointY[index[0]];

        int tmp = 1;
        for (int i = 0; i < 4; i++){
            if (i != index[0] && i != index[1]){
                if (tmp == 1){
                    firstX = pointX[i];
                    firstY = pointY[i];
                } else {
                    thirdX = pointX[i];
                    thirdY = pointY[i];
                }
                tmp++;
            }
        }

        // calculate the vector of parallelogram
        vector1_X = firstX - secondX;
        vector1_Y = firstY - secondY;
        vector2_X = thirdX - secondX;;
        vector2_Y = thirdY - secondY;

        fourthX = secondX + vector1_X + vector2_X;
        fourthY = secondY + vector1_Y + vector2_Y;

        printf("%.3f %.3f\n", fourthX, fourthY);
    }
    return 0;
}
```

UVa10242.cpp hosted with ♡ by GitHub   view raw

Jun 21, 2020

UVa 10642 — Can You Solve It?

題目連結：UVa 10642
難度：一顆星

題目大意：
計算題目所給的兩點之間的移動步數。

解題過程：
1. 觀察出規則：一個點從(0, 0)到 (x, y) 步數為 0+1+…+(x+y)，因此以此為計算基礎，算出 (0, 0) 到 (0, x+y) 的所需步數，再扣除 x，得到 (x, y)步數
2. 算出兩點從 (0, 0)所需之步數再相減，即得解答

使用語言：C++

參考程式如下：

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

int main(){
    int testCase_num;
    int sourceTotal, source_x, source_y;
    int desTotal, destination_x, destination_y;
    long long int step;

    cin >> testCase_num;
    for (int i = 0; i < testCase_num; i++){
        cin >> source_x >> source_y >> destination_x >> destination_y;

        step = 0;
        sourceTotal = source_x + source_y;
        desTotal = destination_x + destination_y;
        for (int j = sourceTotal+1; j <= desTotal; j++){
            step += j;
        }
        step = step - source_x + destination_x;

        cout << "Case " << i+1 << ": " << step << endl;
    }
    return 0;
}
```

UVa10642.cpp hosted with ♡ by GitHub view raw

Jun 21, 2020

UVa 10221 — Satellites

題目連結：UVa 10221
難度：一顆星

題目大意：
給予衛星到地球表面的距離及衛星間的角度，以求最短的弧長及弦長。

解題過程：
1. 先將arc minute 化為角度，並將角度化為最小，以求最短弧長
（若角度>360，則 angle % 360；若角度>180，則 angle = 360-angle）
2. 套用公式，弧長 = 2 * 半徑 * PI * (角度/360)
弦距 = sin(角度) * 半徑 * 2

使用語言：C++

參考程式如下：

```cpp
#define _USE_MATH_DEFINES
#include <iostream>
#include <string>
#include <cmath>
#include <cstdio>
using namespace std;

int main(){
    double distFromSurface, angle;
    double earthRadius = 6440.0; // km
    double degreeOfmin = 360 / 1440;
    double arcDist, chordDist;
    string unit;

    while (cin >> distFromSurface >> angle){
        cin >> unit;

        // 1 degree = 60 arc min
        if (unit == "min"){
            angle = angle / 60;
        }
        while (angle > 360){
            angle -= 360;
        }
        if (angle > 180){
            angle = 360 - angle;
        }

        arcDist = 2 * (distFromSurface + earthRadius) * (angle/360) * M_PI;
        chordDist = sin(angle / 2 * M_PI / 180.0) * (earthRadius+distFromSurface) * 2;

        printf("%.6f %.6f\n", arcDist, chordDist);
    }
    return 0;
}
```

UVa10221.cpp hosted with ♡ by GitHub view raw

Jun 20, 2020

UVa 10908 — Largest Square

題目連結：UVa 10908
難度：一顆星

題目大意：
題目給予 height*width的字元網格，以及幾個(row, column)位置，判斷以 (row, column)為中心為多大的正方形(相同字元構成)。

解題過程：
1. 找出從中心點到邊界的最小距離
2. 用迴圈檢查此距離內存在的最大正方形邊長

使用語言：C++

參考程式如下：

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

int main(){
    int testCaseNum;
    int height, width, squareLocationNum;
    int row, column; // location of square
    int largestSide_half;

    cin >> testCaseNum;
    for(int i = 0; i < testCaseNum; i++){
        cin >> height >> width >> squareLocationNum;
        cout << height << " " << width << " " << squareLocationNum << endl;

        string square[height];
        for (int m = 0; m < height; m++){
            cin >> square[m];
        }

        for (int j = 0; j < squareLocationNum; j++){
            cin >> row >> column;
            largestSide_half = 0; // the length between center and side

            char center = square[row][column];

            int largestHeight = (row < (height-row-1)) ? row : height-row-1;
            int largestWidth = (column < (width-column-1)) ? column : width-column-1;
            int largestPossibleSide_half = (largestHeight < largestWidth) ? largestHeight : larg

            // check if there is x length of a side of square, x = k * 2 + 1
            bool isSquare;
            for (int k = 1; k <= largestPossibleSide_half; k++){
                isSquare = true;
                for (int m = row-k; m <= row+k; m++){
                    for (int n = column-k; n <= column+k; n++){
                        if (square[m][n] != center){
                            isSquare = false;
                            break;
                        }
                    }
                }

                if (isSquare){
                    largestSide_half++;
                } else {
                    break;
                }
            }

            cout << largestSide_half*2+1 << endl;
        }
    }
    return 0;
}
```

UVa10908.cpp hosted with ♡ by GitHub view raw

1

Jun 19, 2020

UVa 11417 — GCD

題目連結：UVa 11417
難度：一顆星

題目大意：
計算 gcd公式得到解答。

解題過程：
1. 按題目所給之計算公式進行計算即可

使用語言：C++

參考程式如下：

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

```cpp
int gcd(int, int);

int main(){
    int n, g;

    while (cin >> n){
        if (n == 0){
            break;
        }

        g = 0;
        for (int i = 1; i < n; i++){
            for (int j = i+1; j <= n; j++){
                g += gcd(i, j);
            }
        }
        cout << g << endl;
    }
    return 0;
}

int gcd(int i, int j){
    int tmp;
    if (j > i){
        tmp = i;
        i = j;
        j = tmp;
    }

    while (j > 0){
        tmp = i % j;
        i = j;
        j = tmp;
    }
    return i;
}
```

UVA11417.cpp hosted with ♡ by GitHub　　view raw

Jun 18, 2020

UVa 10922–2 the 9s

題目連結：UVa 10922

難度：一顆星

題目大意：
用測資各位元相加 **digitSum % 9 = 0** ，判斷測資是否為9的倍數，以及可以遞迴再計算 digitSum 之各位元相加是否為 9 的倍數幾次(為其9-degree)。

解題過程：
1. 測資可能有 1000 bits，因此用 string 存取
2. 用迴圈將測資之各位元相加，並判斷是否為 9 之倍數
3. 若是，則再進入 2. 計算；加總進入迴圈的次數即得 9-degree

使用語言：C++

參考程式如下：

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

int main(){
    string num, tmp;
    int digitSum, degree;

    cin >> num;
    while (num != "0"){
        digitSum = 0;
        degree = 0;

        // check if digitSum is multiple of 9
        for (int i = 0; i < num.length(); i++){
            digitSum = digitSum + (num[i] - '0');
        }

        // if digitSum is multiple of 9, check its 9-degree
        while (digitSum != 0 && digitSum != 9 && digitSum % 9 == 0){
            degree++;
            tmp = to_string(digitSum);
            digitSum = 0;
            for (int i = 0; i < tmp.length(); i++){
                digitSum = digitSum + (tmp[i] - '0');
            }
        }
        if(digitSum == 9){
            degree++;
        }

        if (degree == 0){
            cout << num << " is not a multiple of 9." << endl;
        } else {
            cout << num << " is a multiple of 9 and has 9-degree " << degree << "." << endl;
        }

        cin >> num;
    }
    return 0;
}
```

UVa10922.cpp hosted with ♡ by GitHub view raw

Jun 18, 2020

UVa 10235 — Simply Emirp

題目連結：UVa 10235
難度：一顆星

題目大意：
判斷測資是否為 prime 或 emirp(質數不為迴文，且反轉後仍為質數)。

解題過程：
1. 用迴圈檢查測資是否為質數
2. 若為質數則檢查其反轉數是否等於自己或同為質數

使用語言：C++

參考程式如下：

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

int main(){
    int num;
    bool isPrime, isEmirp;

    while (cin >> num){
        isPrime = true;
        isEmirp = true;

        // Determine if num is a prime number
        for (int i = 2; i <= sqrt(num); i++){
            if (num % i == 0){
                isPrime = false;
                break;
            }
        }

        if (isPrime){
            // reverse the num and determine if reverse_num is a prime
            string tmp = to_string(num);

            int reverse_num = 0;
            for (int i = tmp.length()-1; i >= 0; i--){
                reverse_num = reverse_num * 10 + (tmp[i] - '0');
            }
            if (num == reverse_num){
                isEmirp = false;
            } else {
                for (int i = 2; i <= sqrt(reverse_num); i++){
                    if (reverse_num % i == 0){
                        isEmirp = false;
                        break;
                    }
                }
            }

            if (isEmirp){
                cout << num << " is emirp." << endl;
            } else {
                cout << num << " is prime." << endl;
            }
        } else {
            cout << num << " is not prime." << endl;
        }
    }
    return 0;
}
```

UVa10235.cpp hosted with ♡ by GitHub view raw

Jun 17, 2020

UVa 10190 — Divide, But Not Quite Conquer!

題目連結：UVa 10190
難度：一顆星

題目大意：
判斷測資(n, m) 是否能讓 n 為 m 的完全 k 次方數。

解題過程：
1. 先排除 n<m 或 n ≤1 或 m≤1 ，這些狀況皆為 Boring!
2. n 用迴圈每次除去 m，判斷是否 n % m == 0，若不為 0 則 break 減少計算

使用語言：C++

參考程式如下：

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

int main(){
    long long int n, m;
    vector<long long int> list;
    bool isBoring;

    while (cin >> n >> m){
        isBoring = false;
        list.clear();

        if (n < m || m <= 1 || n <= 1){
            cout << "Boring!" << endl;
        } else {
            for (; n > 1; n /= m){
                if (n % m == 0){
                    list.push_back(n);
                } else {
                    isBoring = true;
                    break;
                }
            }
            list.push_back(1);

            if (!isBoring){
                cout << list[0];
                for (int i = 1; i < list.size(); i++){
                    cout << " " << list[i];
                }
                cout << endl;
            }else{
                cout << "Boring!" << endl;
            }
        }
    }
    return 0;
}
```

UVa10190.cpp hosted with ♡ by GitHub   view raw

Jun 17, 2020

UVa 10812 — Beat the Spread!

題目連結：UVa 10812
難度：一顆星

題目大意：
根據題目提供之兩數 總和 **sum** 及 差 **dif** (取絕對值)，計算出兩數值。

解題過程：
1. 因兩數必為非負整數，以連立方程式思考
2. 若 sum+ dif 不為偶數、sum- dif 不為偶數、sum+ dif < 0、sum- dif < 0，則其解不存在；反之，a = (sum+dif)/2，b = (sum- dif)/2
3. 輸出時，較大者在前

使用語言：C++

參考程式如下：

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

int main(){
    int n;
    long long int sum, dif; // a+b = sum, a-b = dif
    long long int a, b;

    cin >> n;
    for (int i = 0; i < n; i++){
        cin >> sum >> dif;

        a = (sum + dif) / 2;
        b = (sum - dif) / 2;

        if ((sum + dif) % 2 == 1 || (sum - dif) % 2 == 1 || a < 0 || b < 0){
            cout << "impossible" << endl;
        } else {
            cout << a << " " << b << endl;
        }
    }
    return 0;
}
```

UVa10812.cpp hosted with ♡ by GitHub view raw

Jun 17, 2020

UVa 10193 — All You Need Is Love

題目連結：UVa 10193
難度：一顆星

題目大意：
判斷一組 二進位的測資(S1, S2) 是否能夠不斷被同一個數 L 扣除，直到=L
（也就是 S1, S2 是否有 >1 的公因數）

解題過程：
1. 將二進位測資皆轉換為十進位（需使用 **long long int**）
2. 用輾轉相除法來找這對測資的最大公因數

使用語言：C++

參考程式如下：

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

int main(){
    int testCase_num;
    string strA, strB;
    int strA_value, strB_value;

```

```cpp
    cin >> testCase_num;
    for (int i = 0; i < testCase_num; i++){
        cin >> strA >> strB;

        // turn strA and strB into decimal
        int tmp = 1;
        strA_value = 0;
        strB_value = 0;
        for (int j = strA.length()-1; j >= 0; j--){
            strA_value += tmp * (strA[j]- '0');
            tmp *= 2;
        }
        tmp = 1;
        for (int j = strB.length()-1; j >= 0; j--){
            strB_value += tmp * (strB[j] - '0');
            tmp *= 2;
        }

        // find the greatest common divisor of strA and strB
        if (strB_value > strA_value){
            tmp = strA_value;
            strA_value = strB_value;
            strB_value = tmp;
        }
        while (strB_value > 0){
            tmp = strA_value % strB_value;
            strA_value = strB_value;
            strB_value = tmp;
        }

        if (strA_value == 1){
            cout << "Pair #" << (i+1) << ": Love is not all you need!" << endl;
        } else {
            cout << "Pair #" << (i+1) << ": All you need is love!" << endl;
        }
    }
    return 0;
}
```

UVa10193.cpp hosted with ♡ by GitHub view raw

Jun 16, 2020

UVa 10050 — Hartals

題目連結：UVa 10050
難度：一顆星

題目大意：
模擬政黨的罷工頻率(天/次)，來計算損失了幾個工作天。
(星期五、六為休息日不計算在內)

解題過程：
1. 用迴圈模擬出各政黨所有罷工日期，並以 **vector<int>** hartals 記錄
(重複的日期不計入)
2. 再用迴圈將星期五及星期六排除，計算總損失工作天數

使用語言：C++

參考程式如下：

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

int main(){
    int testCase_num;

    cin >> testCase_num;
    for (int i = 0; i < testCase_num; i++){
        int simulationDay;
        int politicalParty_num;
        cin >> simulationDay >> politicalParty_num;

        int partyHartal[politicalParty_num];
        vector<int> hartals;
        for (int j = 0; j < politicalParty_num; j++){
            cin >> partyHartal[j];

            // simulate hartals for all parties
            for (int k = 0; k <= simulationDay; k += partyHartal[j]){
                if (j != 0){
                    // repeated hartals and day0 won't push into vector again
                    if (k != 0 && find(hartals.begin(), hartals.end(), k) == hartals.end()){
                        hartals.push_back(k);
                    }
                } else {
                    if (k != 0){
                        hartals.push_back(k);
                    }
                }
            }
        }

        sort(hartals.begin(), hartals.end());

        int lostWorkingDay = 0;
        for (int j = 0; j < hartals.size(); j++){
            // skip Friday and Saturday
            if (hartals[j] % 7 != 6 && hartals[j] % 7 != 0){
                lostWorkingDay++;
            }
        }
        cout << lostWorkingDay << endl;
    }
    return 0;
}
```

UVa10050.cpp hosted with ♡ by GitHub view raw

Jun 15, 2020

UVa 11005 — Cheapest Base

題目連結：UVa 11005
難度：一顆星

題目大意：
數字以不同 base表示，
使用 “0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZ” 的前 n 個字元；
每個字元有各自的花費 (1-128)，求最便宜的 base 來表示測資數字。

解題過程：
1. 用暴力法，逐個計算從 base 2~128 的所需成本，並將成本存入 vector<int> all_most
2. 從 all_most 中找到 min_cost，並將符合 min_cost 的 base 依序印出

使用語言：C++

參考程式如下：

```cpp
#include <iostream>
#include <vector>
#include <algorithm>
using namespace std;
long long int cal_cost_of_base(int[], long long int, int);

int main(){
    int test_case_num, query_num;
    int cost[36];

    cin >> test_case_num;
    for (int i = 0; i < test_case_num; i++){
        if (i != 0){
            cout << endl;
        }
        cout << "Case " << (i+1) << ":" << endl;

        // access the cost of the 36 characters
        for (int j = 0; j < 36; j++){
            cin >> cost[j];
        }

        cin >> query_num;
        long long int queries[query_num];

        for (int j = 0; j < query_num; j++){
            cin >> queries[j];

            long long int min_cost = 0;
            vector<int> all_cost;

            // calculate all possible cost and find the minimum
            for (int k = 2; k <= 36; k++){
                long long int tmp = cal_cost_of_base(cost, queries[j], k);
                all_cost.push_back(tmp);
                if (min_cost >= tmp || min_cost == 0){
                    min_cost = tmp;
                }
            }

            // print the base of the min_cost
            cout << "Cheapest base(s) for number " << queries[j] << ":";
            for (int k = 0; k < all_cost.size(); k++){
                if (all_cost[k] == min_cost){
                    cout << " " << (k+2);
                }
            }
            cout << endl;
        }
    }
    return 0;
}

// calculate the cost of num being expressed in base
long long int cal_cost_of_base(int cost[], long long int num, int base){
    vector<int> expressed_in_base;
    long long int totalCost = 0;

    // convert num into expressed_in_base
    while (num > 0){
        expressed_in_base.push_back(num % base);
        num /= base;
    }
    reverse(expressed_in_base.begin(), expressed_in_base.end());

    // calculate the cost of expressed_in_base
    for (int i = 0; i < expressed_in_base.size(); i++){
        totalCost += cost[expressed_in_base[i]];
    }

    return totalCost;
}
```

UVa11005.cpp hosted with ♡ by GitHub　　view raw

May 5, 2020

UVa 10931 — Parity

題目連結：UVa 10931
難度：一顆星

題目大意：
計算測資的 parity，定義為測資取二進位中 1的個數 mod 2。

解題過程：
1. 將測資轉為二進位制，且計算其中 ‘1’ 出現的次數

使用語言：C++

參考程式如下：

```
#include <iostream>
#include <string>
```

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

int main(){
    long long int num, parity;
    string str;


    while (cin >> num){
        if (num == 0){
            break;
        }
        str = "";
        while (num > 0){ // turn num into binary representation
            str += to_string(num % 2);
            num /= 2;
        }
        parity = count(str.begin(), str.end(), '1');
        reverse(str.begin(), str.end());

        cout << "The parity of " << str << " is " << parity << " (mod 2)." << endl;
    }

    return 0;
}
```

UVa10931.cpp hosted with ♡ by GitHub　　view raw

May 4, 2020

UVa 10019 — Funny Encryption Method

題目連結：UVa 10019
難度：一顆星

題目大意：
將題目中的加密演算法實作。

解題過程：
1. 將輸入值視為十進位轉為二進位，並計算其中 ‘1’ 的個數為 b1
2. 將輸入值視為十六進位轉為二進位，並計算其中 ‘1’ 的個數為 b2
3. 輸出 b1 及 b2

使用語言：C++

參考程式如下：

```cpp
#include <iostream>
#include <string>
#include <algorithm>
using namespace std;
int dec_to_bin_and_count1(int);
int hex_to_bin_and_count1(int);

int main(){
    int n, m;
    int b1, b2, ans;

    cin >> n;
    for (int i = 0; i < n; i++){
        cin >> m; // the number wants to encrypt

        b1 = dec_to_bin_and_count1(m); // covert m(decimal) to its binary representation
        b2 = hex_to_bin_and_count1(m); // covert m(hexadecimal) to its binary representation
        cout << b1 << " " << b2 << endl;
    }

    return 0;
}

int dec_to_bin_and_count1(int m){
    string x = "";
    int b1, tmp;

    while (m > 0){
        tmp = m % 2;
        m /= 2;
        x += to_string(tmp);
    }
    b1 = count(x.begin(), x.end(), '1');
    return b1;
}

int hex_to_bin_and_count1(int m){
    string x = to_string(m);
    int b2 = 0;

    for (int i = 0; i < x.length(); i++){
        b2 += dec_to_bin_and_count1(x[i] - '0');
    }
    return b2;
}
```

May 4, 2020

UVa 948 — Fibonaccimal Base

題目連結：UVa 948
難度：一顆星

題目大意：
用費氏數列表示十進位數，因為表示方式有很多，所以規定任兩個被選中的項次不能在費氏數列中相鄰。

解題過程：
1. 題目提供測資不超過100,000,000，因此先將費氏數列 (1, 2, 3, 5, …) 存入 **vector<int> fib**中。
2. 將十進位測資與 **vector**中的費氏數列由大至小一一比較，轉換成以 fib 作為進位表示（注意此題不能使用數列中相鄰的項次來表示）

使用語言：C++

參考程式如下：

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

int main(){
    int n, tmp, index = 2;
    int dec_base;
    vector<int> fib;
    string fib_base;

    // store the Fibonacci sequence into the vector
    fib.push_back(1);
    fib.push_back(2);
    while (fib.back() <= 100000000){
        fib.push_back(fib[index-2] + fib[index-1]);
        index++;
    }

    cin >> n;
    for (int i = 0; i < n; i++){
        cin >> dec_base;
        tmp = dec_base;
        index = fib.size() - 1;
        fib_base = ""; // Fibonaccimal base

        while (tmp != 0){
            if (tmp >= fib[index]){
                if (fib_base == "" || fib_base.back() != '1'){ // don't use two consecutive Fibo
                    fib_base = fib_base + "1";
                    tmp = tmp - fib[index];
                }
            } else if (fib_base == ""){

            } else {
                fib_base = fib_base + "0";
            }
            index--;
        }
        for (; index >= 0; index--){
            fib_base = fib_base + "0";
        }
        cout << dec_base << " = " << fib_base << " (fib)" << endl;
    }

    return 0;
}
```

UVa948.cpp hosted with ♡ by GitHub view raw

Apr 29, 2020

UVa 10093 — An Easy Problem!

題目連結：UVa 10093
難度：一顆星

題目大意：
根據題目給的 R 值，找出其最小的 N進位，且 R 一定可以被 (N-1) 整除。

解題過程：

1. 以 string 輸入整數，再用 isalnum() 判斷位元是否為 0–9, A-Z, a-z
2. base 必大於各個位數（例如，1A 的 base必大於 11 ）
3. 將各個位數加總(abc -> a+b+c)，找出最小的 base

使用語言：C++

參考程式如下：

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

int main(){
    string str;
    int index, tmp, base;
    long long int sum;

    while(getline(cin, str)){
        sum = 0;
        base = 1;
        index = 0;

        while (!isalnum(str[index])){
            index++;
        }

        for (int i = index; i < str.length(); i++){
            if (isdigit(str[i])){
                tmp = str[i] - '0';
            } else if (isupper(str[i])){
                tmp = str[i] - 'A' + 10;
            } else if (islower(str[i])){
                tmp = str[i] - 'a' + 36;
            }

            sum += tmp;
            if (base < tmp){
                base = tmp;
            }
        }

        for (;base < 62; base++){
            if (sum % base == 0){
                cout << base + 1 << endl;
                break;
            }
        }
        if (base == 62){
            cout << "such number is impossible!" << endl;
        }
    }
    return 0;
}
```

UVa10093.cpp hosted with ♡ by GitHub view raw

Apr 29, 2020

UVa 10071 — Back to High School Physics

題目連結：UVa 10071
難度：一顆星

題目大意：
在 t 秒之後速度為 v ，求兩倍時間後的位移。

解題過程：
1. 代入公式即可，公式：2*v*t

使用語言：C++

參考程式如下：

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

int main(){
    int v, t;

    while (cin >> v >> t){
        cout << 2 * v * t << endl;
    }
    return 0;
}
```

UVa10071.cpp hosted with ♡ by GitHub view raw

Apr 27, 2020

UVa 11063 — B2-Sequence

題目連結：UVa 11063
難度：一顆星

題目大意：
判斷測資所給的數列是否為 B2-sequence，B2-sequence 的定義為數列中不會有任意兩數總和是相同的，且數列為 1 ≤ b1 < b2 < …… 。

解題過程：
1. 輸入測資時排除 <0 及 沒有由小到大輸入的數列
2. 用一個陣列 **int sum[20001]** 記錄曾出現之兩數相加總和
3. 用迴圈計算 bi + bj，i ≤ j 透過 sum[bi+bj] 檢查此數是否出現過

使用語言：C++

參考程式如下：

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

int main(){
    int n, check, tmp;
    int caseNum = 0;

    while (cin >> n){
        caseNum++;
        check = 1; // initialize check to yes which means this sequence is a B2-seq

        int seq[n];
        for (int i = 0; i < n; i++){
            cin >> seq[i];
            if (seq[i] < 1){
                check = 0;
            }
            if (i != 0 && seq[i-1] >= seq[i]){
                check = 0;
            }
        }
        int sum[20001] = {0}; // use a table to record which num is appeared

        if (check == 1){
            for (int j = 0; j < n; j++){
                for (int k = j; k < n; k++){
                    tmp = seq[j] + seq[k];
                    if (sum[tmp] == 0){
                        sum[tmp] = 1;
                    } else {
                        check = 0;
                        break;
                    }
                }
            }
        }

        if (check == 0){
            cout << "Case #" << caseNum << ": It is not a B2-Sequence." << endl;
        } else {
            cout << "Case #" << caseNum << ": It is a B2-Sequence." << endl;
        }
        cout << endl;
    }
    return 0;
}
```

UVa11063.cpp hosted with ♡ by GitHub view raw

Apr 27, 2020

UVa 11461 — Square Numbers

題目連結：UVa 11461
難度：一顆星

題目大意：
給定兩個整數 a, b，找到 a 和 b 之間的完全平方數個數。

解題過程：
1. 計算 a 及 b 之平方數，並相減算出其個數
（需考慮 a 本身是否為完全平方數，所以用 **ceil()**計算）

使用語言：C++

參考程式如下：

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

int main (){
    int a, b;
    int a_sqrt, b_sqrt, count;

    while (cin >> a >> b){
        if (a == 0 && b == 0){
            break;
        }

        a_sqrt = (int)ceil(sqrt(a));
        b_sqrt = (int)sqrt(b);
        count = b_sqrt - a_sqrt + 1;

        cout << count << endl;
    }
    return 0;
}
```

UVa11461.cpp hosted with ♡ by GitHub view raw

Apr 14, 2020

UVa 11349 — Symmetric Matrix

題目連結：UVa 11349
難度：一顆星

題目大意：
檢查題目所給之矩陣是否對稱 (為非負，且相對於中心點對稱)。

解題過程：
1. 處理輸入 “N = _” (記得 cin.ignore() 清除緩存)
2. 檢查矩陣元素是否為非負整數
3. 將矩陣元素按順序排成一列後，檢查是否迴文來判斷是否對稱於中心點

使用語言：C++

參考程式如下：

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

int main(){
    int t; // size of test cases
    int n; // the dimension of square matrix

    cin >> t;
    for (int a = 0; a < t; a++){
        cin.ignore();
        scanf("N = %d", &n);

        long long int matrix[n*n];
        int check_neg = 1; // if matrix negative
        int check_sym = 1; // if matrix symmetric

        for (int i = 0; i < n*n; i++){
            cin >> matrix[i];
            if (matrix[i] < 0){
                check_neg = 0;
            }
        }

        if (check_neg != 0){
            for (int i = 0; i < n*n/2; i++){
                if (matrix[i] != matrix[n*n-i-1]){
                    check_sym = 0;
                    break;
                }
            }
        }

        if (check_sym && check_neg){
            cout << "Test #" << a+1 << ": Symmetric." << endl;
        } else {
            cout << "Test #" << a+1 << ": Non-symmetric." << endl;
        }
    }
    return 0;
}
```

UVa11349.cpp hosted with ♡ by GitHub view raw

Apr 13, 2020

UVa 10783 — Odd Sum

題目連結：UVa 10783
難度：一顆星

題目大意：
計算題目提供兩數範圍內之奇數總和。

解題過程：
1. 用迴圈計算 a~b之奇數相加

使用語言：C++

參考程式如下：

```cpp
#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;
}
```

Apr 12, 2020

UVa 10268–498-bis

題目連結：UVa 10268
難度：一顆星

題目大意：
根據題目提供資訊計算出多項式微分後的值。

解題過程：
1. 處理輸入值，注意輸入之兩個係數間可能含多個空白、不知係數數量未知
2. 使用 **long long int** 型別，依算式計算出微分答案 (注意 x = 0 之結果)
3. 使用函式庫之 **pow()**會有誤差

使用語言：C++

參考程式如下：

```
 #include <iostream>
#include <string>
#include <cstdlib>
using namespace std;

int main(){
    int x, index;
    long long int ans, exp;
    long long int a[1000000];
    string s, tmp;

    while (cin >> x){
        cin.ignore();
        getline(cin, s);

        tmp = "";
        index = 0;
        ans = 0;
        exp = 1;

        for (int i = 0; i < s.length(); i++){ // input data processing
            if (s[i] != ' '){
                tmp += s[i];
            } else {
                a[index] = atoi(tmp.c_str());
                index++;
                tmp = "";
            }
        }
        a[index] = atoi(tmp.c_str());

        if (x == 0){
            ans = a[index-1];
        } else {
            for (int i = index-1; i >= 0; i--){
                ans += a[i] * (index - i) * exp;
                exp *= x;
            }
        }
        cout << ans << endl;
    }

    return 0;
}
```

UVa10268.cpp hosted with ♡ by GitHub view raw

1

Apr 11, 2020

UVa 10170 — The Hotel with Infinite Rooms

題目連結：UVa 10170
難度：一顆星

題目大意：
題目提供 第一天入住之人數 s 及 日期 n，欲求第 n 天入住之人數為多少。
入住團體以人數為住宿天數，並且下一團之人數為上一團人數+1。

解題過程：
1. 因題目天數範圍為 $1\sim10^{15}$，所以使用 **long int** 型別
2. 用迴圈計算，將目前天數加上目前團體人數得出下一團所住之起始日期，也將目前團體人數+1 得出下團人數；直到達成題目條件日期

使用語言：C++

參考程式如下：

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

```cpp

int main(){
    int s; // size of the group
    long int n; // the group staying in the hotel on D-th day
    long int peopleNum, day;

    while (cin >> s >> n){
        peopleNum = s;
        day = 1;

        while (n >= day){
            day = day + peopleNum;
            peopleNum++;
        }
        peopleNum--;
        cout << peopleNum << endl;
    }

    return 0;
}
```

UVa10170.cpp hosted with ♡ by GitHub view raw

Apr 10, 2020

UVa 10056 — What is the Probability ?

題目連結：UVa 10056
難度：一顆星

題目大意：
透過輸入值計算第 i 個人達成特定事件的機率。
r：回合數，n：總人數，p：成功機率，q = (1- p)

第 R回合成功機率：q^[(r-1)*n] * q^(i-1) * p

公式：q^(i-1) * p / (1- q^n)

解題過程：
1. 將輸入值代入公式中 (p 使用 **double** 型別)
2. 注意輸出格式要準確到小數點後 4位，並且當 p=0 時，輸出即為 0.0000

使用語言：C++

參考程式如下：

```cpp
#include <iostream>
#include <cmath>
#include <iomanip>
using namespace std;

int main(){
    int s, n, i_th;
    double p; // probability of successful event
    double q, ans;

    cin >> s;
    for (int i = 0; i < s; i++){
        cin >> n >> p >> i_th;

        if (p == 0){
            ans = 0;
        } else {
            q = 1 - p;
            ans = pow(q, i_th-1) * p / (1 - pow(q, n));
        }
        cout << fixed << setprecision(4) << ans << endl;
    }
    return 0;
}
```

UVa10056.cpp hosted with ♡ by GitHub view raw

Apr 9, 2020

UVa 10038 — Jolly Jumpers

題目連結：UVa 10038
難度：一顆星

題目大意：
判斷輸入資料是否為 **jolly jumper**，其定義為在一串個數為 n 的序列中連續元素的差之絕對值包含了 1 到 n-1 所有值

解題過程：
1. 若序列只有一個數字必為 jolly jumper
2. 用 int[] 記錄連續輸入值的差之絕對值
3. 迴圈檢查陣列中是否有 1~n-1 未出現的值

使用語言：C++

參考程式如下：

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

int main(){
    int n, tmp;

    while (cin >> n){
        if (n == 1){
            cin >> tmp;
            cout << "Jolly" << endl;
        } else {
            int seq[n];
            int check[n]; // check if 1~n-1 appeared

            for (int i = 0; i < n; i++){ // initially
                check[i] = 0;
            }

            for (int i = 0; i < n; i++){
                cin >> seq[i];

                if (i != 0){
                    tmp = abs(seq[i-1] - seq[i]);
                    check[tmp] = 1;
                }
            }

            for (int i = 1; i < n; i++){
                if (check[i] == 0){
                    cout << "Not jolly" << endl;
                    break;
                }
                if (i == n-1){
                    cout << "Jolly" << endl;
                }
            }
        }
    }

    return 0;
}
```

UVa10038.cpp hosted with ♡ by GitHub view raw

Apr 7, 2020

UVa 12019 — Doom’s Day Algorithm

題目連結：UVa 12019
難度：一顆星

題目大意：
透過題目給予的日期及其星期，求出其他輸入日期為星期幾。

解題過程：
1. 將題目給的資訊及星期存入 **int[]** 及 **string[]** 中
2. 依輸入值與題目資訊之月份比對，求出星期幾

使用語言：C++

參考程式如下：

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

int main(){
    int n;
    int month, day, weekDay;
    string week[7] = {"Monday", "Tuesday", "Wednesday", "Thursday", "Friday", "Saturday", "Sunda

    // 2011 1/10, 2/21, 4/4, 5/9, 6/6, 7/11,
    // 8/8, 9/5, 10/10 , 11/7 and 12/12 are Mondays.
    // 3/1 is Tuesday
    int mondays[12] = {10, 21, 0, 4, 9, 6, 11, 8, 5, 10, 7, 12};

    cin >> n;
    for (int i = 0; i < n; i++){
        cin >> month >> day;

        month--; // to fit the index of array
        month--; // to fit the index of array

        weekDay = (day - mondays[month] + 35) % 7;

        cout << week[weekDay] << endl;
    }
    return 0;
}
```

UVa12019.cpp hosted with ♡ by GitHub view raw

Apr 6, 2020

UVa 272 — TEX Quotes

題目連結：UVa 272
難度：一顆星

題目大意：
轉換一般 " 為一對雙引號 “” ，用 `` 及 '' 表示。

解題過程：
1. 題目不含巢狀雙引號，因此用 **int** 計算其出現次數，奇數為左雙引號，偶數為右雙引號

使用語言：C++

參考程式如下：

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

int main(){
    string tmp;
    int length;
    int count = 0;

    while (getline(cin, tmp)){
        length = tmp.length();

        for (int i = 0; i < length; i++){
            if (tmp[i] == '\"'){
                count++;

                if (count % 2 == 1){ // left-double-quote
                    cout << "``";
                } else {
                    cout << "''"; // right-double-quote
                }
            } else {
                cout << tmp[i];
            }
        }
        cout << endl;
    }
    return 0;
}
```

Apr 6, 2020

UVa 490 — Rotating Sentences

題目連結：UVa 490
難度：一顆星

題目大意：
將原先「由左往右，再由上至下」輸入的文字，改為直行輸出（由上往下，再由右至左）。

解題過程：
1. 將所有輸入值存入 **string[]**，並找出最長字串長度
（注意：宣告陣列大小時要超過100，否則會runtime error）
2. 依順序改為直行輸出，記得補足空白

使用語言：C++

參考程式如下：

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

```cpp

int main(){
    string s[101];
    int index = 0;
    int max = 0;

    while (getline(cin, s[index])) {
        if (max < s[index].length()){ // find out the max length of strings
            max = s[index].length();
        }
        index++;
    }

    for (int i = 0; i < max; i++){
        for (int j = index-1; j >= 0; j--){
            if (s[j].length() > i){
                cout << s[j][i];
            } else {
                cout << ' ';
            }
        }
        cout << endl;
    }
    return 0;
}
```

Apr 4, 2020

UVa 10252 — Common Permutation

題目連結：UVa 10252
難度：一顆星

題目大意：
給予兩個小寫字串，找出重複之字母。

解題過程：
1. 先計算出兩個字串之各個字母出現次數
（注意：可能包含空字串，使用**getline()**，而非 **cin>>**）
2. 比較兩者次數，取最小者，存入 char letter[26]
3. 依字母順序印出字母的個數

使用語言：C++

參考程式如下：

```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;
}
```

UVa10252.cpp hosted with ♡ by GitHub view raw

1

Mar 30, 2020

UVa 11332 — Summing Digits

題目連結：UVa 11332
難度：一顆星

題目大意：
將輸入值之所有位數相加，直到相加至 <10 之正整數，即輸出。

解題過程：
1. 因為想快速拆解各個位數，將輸入值存為 **string**來運算
2. 若輸入值為個位數，直接輸出；否則，不斷將每次取得之總和的各個位數拆開後相加

使用語言：C++

參考程式如下：

```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;
}
```

UVa11332.cpp hosted with ♡ by GitHub view raw

Mar 30, 2020

UVa 10222 — Decode the Mad man

題目連結：UVa 10222
難度：一顆星

題目大意：
要解開教授說話的密碼，以其所給的符號或字母在鍵盤上的位置往右兩位來尋找正確解碼。

解題過程：
1. 將鍵盤內容按順序由左往右存為 string
2. 依序判斷輸入值之密碼轉換，注意空白及大小寫變化

使用語言：C++

參考程式如下：

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

int main(){
    string keyboard = "`1234567890-=qwertyuiop[]\\asdfghjkl;'zxcvbnm,./";
    string input;
    int index = 0;

    getline(cin, input);
    transform(input.begin(), input.end(), input.begin(), ::tolower);

    for (int i = 0; i < input.length(); i++){
        if (input[i] == ' '){
            cout << " ";
            continue;
        }
        index = keyboard.find(input[i])-2;
        cout << keyboard[index];
    }
    cout << endl;
    return 0;
}
```

UVa10222.cpp hosted with ♡ by GitHub view raw

Mar 29, 2020

UVa 10008 — What’s Cryptanalysis?

題目連結：UVa 10008
難度：一顆星

解題過程：
1. 先將所有輸入值轉換成小寫，接續放入一個 **string**
2. 將 string **排序**減少計算各個字母次數的時間，並以 **letter[]** 記錄
3. 用 **distance(max_element())** 找出 letter[] 中最大值之 index，並印出

使用語言：C++

參考程式如下：

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

int main(){
    int n, temp;
    int letter[26] = {0};
    int maxIndex;
    char maxLetter;
    string str = " ";
    string tmpStr;

    cin >> n;
    tmpStr = "\n";

    for (int i = 0; i <= n; i++){
        getline(cin, tmpStr);
        transform(tmpStr.begin(), tmpStr.end(), tmpStr.begin(), ::tolower); // transfer upper le
```

```
            str += tmpStr;
        }

        sort(str.begin(), str.end());

        for (int i = 0; i < str.length(); i++){
            if (str[i] < 97 || str[i] > 122){ // str[i][j] is not letter(a~z)
                continue;
            }
            temp = (int)str[i] - 97;
            letter[temp]++;
        }

        for (int i = 0; i < 26; i++){
            maxIndex = max_element(letter, letter+26) - letter;
            maxLetter = 'A' + maxIndex;
            if (letter[maxIndex] == 0){
                break;
            }
            cout << maxLetter << " " << letter[maxIndex] << endl;
            letter[maxIndex] = 0;
        }

        return 0;
    }
```

UVa10008.cpp hosted with ♡ by GitHub　　view raw

Mar 26, 2020

UVa 10420 — List of Conquests

題目連結：UVa 10420
難度：一顆星

解題過程：
1. 先將輸入值放入 **string[]** 中
2. 使用 **sort()** 函數將字串依字母順序排序
3. 用迴圈計算重複出現之國家次數
（注意：若先計算次數再排序會浪費很多時間，造成 Time limit exceeded）

使用語言：C++

參考程式如下：

```cpp
#include <iostream>
#include <string>
#include <algorithm>
using namespace std;
int compare(const void *, const void *);

int main(){
    int n, count, j;
    string name;

    cin >> n;
    string countries[n];

    for (int i = 0; i < n; i++){
        cin >> countries[i];
        getline(cin, name);
    }

    sort(countries, countries+n);

    for (int i = 0; i < n;){
        cout << countries[i] << " ";
        count = 1;

        for (j = i+1; j < n; j++){
            // countries[] has been sorted,
            // so if c[i] != c[j] present that it won't have the same country after c[j]
            if (countries[i] != countries[j]){
                break;
            }
            count++;
        }
        i = j;
        cout << count << endl;
    }
    return 0;
}
```

UVa10420.cpp hosted with ♡ by GitHub   view raw

Mar 23, 2020

UVa 10101 — Bangla Numbers

題目連結：UVa 10101
難度：一顆星

解題過程：
1. 因輸入值小於 999,999,999,999,999(15位數)，因此使用 **long long int**進行判斷
2. 將輸入值分為 前8位數 及 後8位數進行文本轉換
（注意：輸出格式，若為 0 則不須 print，可參考 udebug ）
3. output 要控制寬度，使用 **setw()** 函式

使用語言：C++

參考程式如下：

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

const int kuti = 10000000;
const int lakh = 100000;
const int hajar = 1000;
const int shata = 100;
void transfer(int, int);

int main() {
    long long int num;
    int firstHalf, secondHalf;
    int count = 0;

    while (cin >> num) {
        count++;

        cout << setw(4) << setfill(' ') << count << ". ";

        if (num == 0){
            cout << "0";
        } else if (num >= kuti) {
            firstHalf = num / kuti;
            secondHalf = num % kuti;
            transfer(firstHalf, 0);
            cout << " kuti";
            transfer(secondHalf, 1);
        } else {
            transfer(num, 0);
        }
        cout << endl;
    }
    return 0;
}

void transfer(int n, int count){
    int k, l, h, s, x;

    l = n / lakh;
    if (l > 99){
        k = l / 100;
        cout << k << " kuti";
        l = l % 100;
        count++;
    }
    if (l != 0){
        if (count != 0){
            cout << " ";
        }
        cout << l << " lakh";
        count++;
    }
    n = n % lakh;
    h = n / hajar;
    if (h != 0) {
        if (count != 0){
            cout << " ";
        }
        cout << h << " hajar";
        count++;
    }
    n = n % hajar;
    s = n / shata;
    if (s != 0) {
        if (count != 0){
            cout << " ";
        }
        cout << s << " shata";
        count++;
    }
    x = n % shata;
    if (x != 0){
        if (count != 0){
            cout << " ";
        }
        cout << x;
    }
}
```

UVa10101.cpp hosted with ♡ by GitHub view raw

1

Mar 20, 2020

UVa 10929 — You can say 11

題目連結：UVa 10929
難度：一顆星

解題過程：
1. 因為 input值可能超過 1000位數，使用 **string** 判斷是否為 11的倍數
2. string 轉換為數字：將 (int)str[x]-48

使用語言：C++

參考程式如下：

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

int main(){
    string num;
    int oddSum, evenSum, length;

    while (cin >> num){
        if (num == "0"){
            break;
        }

        oddSum = 0;
        evenSum = 0;
        length = num.length();

        for (int i = 0; i < length-1; i+=2){  // count the sum of every odd and even bit
            evenSum = evenSum + (int)num[i] - 48; // change str[] to int
            oddSum = oddSum + (int)num[i+1] - 48;
        }
        if (length % 2 == 1){
            evenSum = evenSum + (int)num[length-1] - 48;
        }

        if ((oddSum - evenSum) % 11 == 0){
            cout << num << " is a multiple of 11.\n";
        } else {
            cout << num << " is not a multiple of 11.\n";
        }
    }
    return 0;
}
```

UVa10929.cpp hosted with ♡ by GitHub view raw

Mar 18, 2020

UVa 10035 — Primary Arithmetic

題目連結：UVa 10035
難度：一顆星

解題過程：
1. 計算輸入值的各個**位數**相加是否進位，若進位記得將加入下一個位數計算（注意：輸入值使用 unsigned long long int）
2. 判斷各輸出格式
（注意：carry 超過 1 時，operation 要加s）

使用語言：C++

參考程式如下：

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

int main(){
    unsigned long long int a, b;
    int digit_a, digit_b, carry, c;

    while (cin >> a >> b){
        carry = 0;
        c = 0;

        if (a == 0 && b == 0){
            break;
        }

        while (a > 0 || b > 0){
            digit_a = a % 10;
            a = a / 10;
            digit_b = b % 10;
            b = b / 10;
```

```cpp
            if (digit_a + digit_b + c >= 10){
                carry++;
                c = 1;
            } else {
                c = 0;
            }
        }

        if (carry == 0){
            cout << "No carry operation." << endl;
        } else if (carry > 1) {
            cout << carry << " carry operations." << endl;
        } else {
            cout << "1 carry operation." << endl;
        }
    }

    return 0;
}
```

UVa10035.cpp hosted with ♡ by GitHub view raw

17

Mar 18, 2020

UVa 10055 — Hashmat the Brave Warrior

題目連結：UVa 10055
難度：一顆星

解題過程：
1. 輸入值之兩數取絕對值即可
（注意：輸入值大小 ≤ 2^{32}，使用 long long int 型別）

使用語言：C++

參考程式如下：

```cpp
#include <iostream>
#include <stdlib.h>
using namespace std;

int main(){
    long long int h_army, op_army, dif;
    while(cin >> h_army >> op_army){
        dif = abs(h_army - op_army);
        cout << dif << endl;
    }

    return 0;
}
```

UVa10055.cpp hosted with ♡ by GitHub view raw

Mar 18, 2020

UVa 10041 — Vito's Family

題目連結：UVa 10041
難度：一顆星

解題過程：
1. 宣告一個 **array** 放親戚家門牌號碼
2. 將門牌號碼由小到大排序，找出中位數為 Vito的住處
（注意：在直線上找出一點到其他點的距離和最小，表示此點為其他點座標的中位數）
3. 計算 Vito家到其他人家的距離

使用語言：C++

參考程式如下：

```
#include <iostream>
#include <algorithm>
#include <cmath>
using namespace std;

int main(void){
    int case_num, rel_num, median, sum;
    int addr[500];

    cin >> case_num;
    for (int i = 0; i < case_num; i++){
        median = 0;
        sum = 0;
        cin >> rel_num;
```

```cpp

        for (int j = 0; j < rel_num; j++){
            cin >> addr[j];
        }

        // Find the median of addr[]
        sort(addr, addr + rel_num);
        if (rel_num % 2 == 1){
            median = addr[rel_num / 2];
        } else {
            median = (addr[rel_num / 2 - 1] + addr[rel_num / 2]) / 2;
        }

        // compute the sum of distance to each one of relatives
        for (int j = 0; j < rel_num; j++){
            sum = sum + abs(addr[j] - median);
        }
        cout << sum << "\n";
    }
    return 0;
}
```

UVa10041.cpp hosted with ♡ by GitHub view raw

Mar 18, 2020

UVa 100 — The 3n + 1 problem

題目連結：UVa 100
難度：一顆星

解題過程：
1. 寫出題目給的演算法
2. 用 **for** 迴圈計算 i、j 兩數間的所有數之 cycle-length，並記錄 max值
（注意：輸入值 i 及 j 不一定為 i ≤ j）
3. 用 **while** 迴圈使程式可重複輸入

使用語言：C++

參考程式如下：

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

```

int main(void){
    int i, j, time, temp, max, start, end;

    while(cin >> i >> j){
        max = -1;

        //check if i <= j
        if (i > j){
            start = j;
            end = i;
        }else{
            start = i;
            end = j;
        }

        for (int k = start; k <= end; k++){
            time = 0;
            temp = k;

            while (temp != 1){
                if (temp % 2 == 1){    // temp is odd
                    temp = 3 * temp + 1;
                } else {  // temp is even
                    temp = temp / 2;
                }
                time++;
            }
            time++;

            if (max < time){
                max = time;
            }
        }
        cout << i << " " << j << " " << max << "\n";
    }
    return 0;
}
```

UVa100.cpp hosted with ♡ by GitHub view raw