Contents

1 Section1

1.1 limits

```
/*
===================================== LIMITS ======================================
[Type]            [size]  [range]
char                1      127 to -128
signed char         1      127 to -128
unsigned char       1      0 to 255
short               2      32767 to -32768
int                 4      2147483647 to -2147483648
unsigned int        4      0 to 4294967295
long                4      2147483647 to -2147483648
unsigned long       4      0 to 18446744073709551615
long long           8      9223372036854775807 to -9223372036854775808
double              8      1.79769e+308 to 2.22507e-308
long double         16     1.18973e+4932 to 3.3621e-4932
float               4      3.40282e+38 to 1.17549e-38
unsigned long long  8      18446744073709551615
string              32
=============================== Printable characters ================================
int     char    int     char    int     char
32        64    @   96    `
33    !   65    A   97    a
34      "   66    B   98    b
35    #   67    C   99    c
36      $   68    D   100   d
37    %   69    E   101   e
38    &   70    F   102   f
39    '   71    G   103   g
40    (   72    H   104   h
41    )   73    I   105   i
42    *   74    J   106   j
43    +   75    K   107   k
44    ,   76    L   108   l
45    -   77    M   109   m
46    .   78    N   110   n
47    /   79    O   111   o
48    0   80    P   112   p
49    1   81    Q   113   q
50    2   82    R   114   r
51    3   83    S   115   s
52    4   84    T   116   t
53    5   85    U   117   u
54    6   86    V   118   v
55    7   87    W   119   w
56    8   88    X   120   x
57    9   89    Y   121   y
58    :   90    Z   122   z
59    ;   91    [   123   {
60    <   92    \   124   |
61    =   93    ]   125   }
62    >   94    ^   126   ~
63    ?   95    _

==========================================================
*/

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

class Node {
public:
    int val;
    vector<Node*> children;

    Node() {}

    Node(int _val) {
        val = _val;
    }

    Node(int _val, vector<Node*> _children) {
        val = _val;
        children = _children;
    }
};

struct ListNode {
    int val;
    ListNode *next;
    ListNode() : val(0), next(nullptr) {}
    ListNode(int x) : val(x), next(nullptr) {}
    ListNode(int x, ListNode *next) : val(x), next(next) {}
};

struct TreeNode {
    int val;
    TreeNode *left;
    TreeNode *right;
    TreeNode() : val(0), left(nullptr), right(nullptr) {}
    TreeNode(int x) : val(x), left(nullptr), right(nullptr) {}
    TreeNode(int x, TreeNode *left, TreeNode *right) : val(x), left(left), right(right) {}
};

class ListProblem {
    vector<int> nums={};
public:
    void solve() {
        return;
    }

    ListNode* buildList(int idx) {
        if(idx == nums.size()) return NULL;
        ListNode *current=new ListNode(nums[idx++],current->next);
        return current;
    }

    void deleteList(ListNode* root) {
        if(root == NULL) return;
        deleteList(root->next);
        delete root;
        return;
    }
};

class TreeProblem {
    int null = INT_MIN;
    vector<int> nums = {}, result;
public:
    void solve() {

        return;
    }

    TreeNode* buildBinaryTreeUsingDFS(int left, int right) {
        if((left > right) || (nums[(left+right)/2] == null)) return NULL;
        int mid = (left+right)/2;
        TreeNode* current = new TreeNode(
            nums[mid],
            buildBinaryTreeUsingDFS(left,mid-1),
            buildBinaryTreeUsingDFS(mid+1,right));
        return current;
    }

    TreeNode* buildBinaryTreeUsingBFS() {
        int idx = 0;
        TreeNode* root = new TreeNode(nums[idx++]);
        queue<TreeNode*> q;
        q.push(root);
        while(idx < nums.size()) {
            if(nums[idx] != null) {
                TreeNode* left = new TreeNode(nums[idx]);
                q.front()->left = left;
                q.push(left);
            }
            idx++;
            if((idx < nums.size()) && (nums[idx] != null)) {
                TreeNode* right = new TreeNode(nums[idx]);
                q.front()->right = right;
                q.push(right);
            }
            idx++;
            q.pop();
        }
        return root;
    }

    Node* buildNAryTree() {
        int idx = 2;
        Node *root = new Node(nums.front());
        queue<Node*> q;
        q.push(root);
        while(idx < nums.size()) {
            while((idx < nums.size()) && (nums[idx] != null)) {
                Node *current = new Node(nums[idx++]);
                q.front()->children.push_back(current);
                q.push(current);
            }
            idx++;
            q.pop();
        }
        return root;
    }

    void deleteBinaryTree(TreeNode* root) {
        if(root->left != NULL) deleteBinaryTree(root->left);
        if(root->right != NULL) deleteBinaryTree(root->right);
        delete root;
        return;
    }

    void deleteNAryTree(Node* root) {
        if(root == NULL) return;
        for(int i=0; i<root->children.size(); i++) {
            deleteNAryTree(root->children[i]);
            delete root->children[i];
        }
        delete root;
        return;
    }

    void inorderTraversal(TreeNode* root) {
        if(root == NULL) return;
        inorderTraversal(root->left);
        cout<<root->val<<' ';
        inorderTraversal(root->right);
        return;
    }
};

int main() {

    return 0;
}
```

2 Section2

2.1 thm

- 中文測試
- $\sum_{i=1}^{n} i^2 = \frac{n(n+1)(2n+1)}{6}$