

Contents

1	Section1	1
1.1	limits	1
2	Section2	2
2.1	thm	2

1 Section1

1.1 limits

```

1  /*
2  ===== LIMITS
3  =====
4  [Type]          [size] [range]
5  char            1      127 to -128
6  signed char     1      127 to -128
7  unsigned char   1      0 to 255
8  short           2      32767 to -32768
9  int              4      2147483647 to -2147483648
10 unsigned int     4      0 to 4294967295
11 long             4      2147483647 to -2147483648
12 unsigned long    4      0 to 18446744073709551615
13 long long        8      9223372036854775807 to
    -9223372036854775808
14 double           8      1.79769e+308 to
    2.22507e-308
15 long double      16     1.18973e+4932 to
    3.3621e-4932
16 float            4      3.40282e+38 to 1.17549e-38
17 unsigned long long 8      18446744073709551615
18 string           32
19 ===== Printable characters
20 =====
21 int      char      int      char      int      char
22 32        !         64        @         96        `
23 33        "         65        A         97        a
24 34        #         66        B         98        b
25 35        $         67        C         99        c
26 36        %         68        D        100        d
27 37        &         69        E        101        e
28 38        '         70        F        102        f
29 39        (         71        G        103        g
30 40        (         72        H        104        h
31 41        )         73        I        105        i
32 42        *         74        J        106        j
33 43        +         75        K        107        k
34 44        ,         76        L        108        l
35 45        -         77        M        109        m
36 46        .         78        N        110        n
37 47        /         79        O        111        o
38 48        0         80        P        112        p
39 49        1         81        Q        113        q
40 50        2         82        R        114        r
41 51        3         83        S        115        s
42 52        4         84        T        116        t
43 53        5         85        U        117        u
44 54        6         86        V        118        v
45 55        7         87        W        119        w
46 56        8         88        X        120        x
47 57        9         89        Y        121        y
48 58        :         90        Z        122        z
49 59        ;         91        [        123        {
50 60        <         92        \        124        |
51 61        =         93        ]        125        }
52 62        >         94        ^        126        ~
53 63        ?         95        _
54
55 =====
56 */
57 #include<bits/stdc++.h>
    using namespace std;

```

```

58
59 class Node {
60 public:
61     int val;
62     vector<Node*> children;
63
64     Node() {}
65
66     Node(int _val) {
67         val = _val;
68     }
69
70     Node(int _val, vector<Node*> _children) {
71         val = _val;
72         children = _children;
73     }
74 };
75
76 struct ListNode {
77     int val;
78     ListNode *next;
79     ListNode() : val(0), next(nullptr) {}
80     ListNode(int x) : val(x), next(nullptr) {}
81     ListNode(int x, ListNode *next) : val(x),
        next(next) {}
82 };
83
84 struct TreeNode {
85     int val;
86     TreeNode *left;
87     TreeNode *right;
88     TreeNode() : val(0), left(nullptr),
        right(nullptr) {}
89     TreeNode(int x) : val(x), left(nullptr),
        right(nullptr) {}
90     TreeNode(int x, TreeNode *left, TreeNode *right)
        : val(x), left(left), right(right) {}
91 };
92
93 class ListProblem {
94     vector<int> nums={};
95 public:
96     void solve() {
97         return;
98     }
99
100     ListNode* buildList(int idx) {
101         if(idx == nums.size()) return NULL;
102         ListNode *current=new
            ListNode(nums[idx++],current->next);
103         return current;
104     }
105
106     void deleteList(ListNode* root) {
107         if(root == NULL) return;
108         deleteList(root->next);
109         delete root;
110         return;
111     }
112 };
113
114 class TreeProblem {
115     int null = INT_MIN;
116     vector<int> nums = {}, result;
117 public:
118     void solve() {
119
120         return;
121     }
122
123     TreeNode* buildBinaryTreeUsingDFS(int left, int
        right) {
124         if((left > right) || (nums[(left+right)/2] ==
            null)) return NULL;
125         int mid = (left+right)/2;
126         TreeNode* current = new TreeNode(
            nums[mid],

```

```

128         buildBinaryTreeUsingDFS(left,mid-1),
129         buildBinaryTreeUsingDFS(mid+1,right));
130     return current;
131 }
132
133 TreeNode* buildBinaryTreeUsingBFS() {
134     int idx = 0;
135     TreeNode* root = new TreeNode(nums[idx++]);
136     queue<TreeNode*> q;
137     q.push(root);
138     while(idx < nums.size()) {
139         if(nums[idx] != null) {
140             TreeNode* left = new
141                 TreeNode(nums[idx]);
142             q.front()->left = left;
143             q.push(left);
144         }
145         idx++;
146         if((idx < nums.size()) && (nums[idx] !=
147             null)) {
148             TreeNode* right = new
149                 TreeNode(nums[idx]);
150             q.front()->right = right;
151             q.push(right);
152         }
153         idx++;
154         q.pop();
155     }
156     return root;
157 }
158
159 Node* buildNAryTree() {
160     int idx = 2;
161     Node *root = new Node(nums.front());
162     queue<Node*> q;
163     q.push(root);
164     while(idx < nums.size()) {
165         while((idx < nums.size()) && (nums[idx]
166             != null)) {
167             Node *current = new Node(nums[idx++]);
168             q.front()->children.push_back(current);
169             q.push(current);
170         }
171         idx++;
172         q.pop();
173     }
174     return root;
175 }
176
177 void deleteBinaryTree(TreeNode* root) {
178     if(root->left != NULL)
179         deleteBinaryTree(root->left);
180     if(root->right != NULL)
181         deleteBinaryTree(root->right);
182     delete root;
183     return;
184 }
185
186 void deleteNAryTree(Node* root) {
187     if(root == NULL) return;
188     for(int i=0; i<root->children.size(); i++) {
189         deleteNAryTree(root->children[i]);
190         delete root->children[i];
191     }
192     delete root;
193     return;
194 }
195
196 void inorderTraversal(TreeNode* root) {
197     if(root == NULL) return;
198     inorderTraversal(root->left);
199     cout<<root->val<<' ';
200     inorderTraversal(root->right);
201     return;
202 }
203
204 };

```

```

199 int main() {
200
201     return 0;
202 }

```

2 Section2

2.1 thm

· 中文測試

$$\cdot \sum_{i=1}^n i^2 = \frac{n(n+1)(2n+1)}{6}$$