

□□□□

 alt text □□□□□□□□□□ □□□□□□□□□□□□□□□□□□

```
#include<bits/stdc++.h>
using namespace std;
int main()
{
    int n;
    int flag = 0;
    cin >> n;
    vector<string> temp[n] , s;
    string str;
    string target;
    for(int i = 0; i < n; i++)
    {
        cin >> str;
        temp[i] = str;
        str = NULL ;
    }
    cin >> target;
    for(int i = 0; i < n; i++)
    {
        if(temp[i] == target)
        {
            s.push_back(temp[i]);
            flag = 1;
            break;
        }
        if(temp[i] == "return")
            s.pop_back(temp[i]);
        else
            s.push_back(temp[i]);
    }
    if(flag)
    {
        for(int i = 0; i < s.size();i++)
        {
            if(i) cout << "->"
                cout << s[i];
        }
        cout << endl;
    }
    else
    {
        cout << "NOT REFERENCED"
    }
}
return 0;
}
```

队列 队列的应用

队列应用  alt text

- 队列的应用
- 队列的应用
 - 队列的应用
 - 队列的应用
 - 队列的应用
 - 队列的应用
- 队列的应用

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

int min_num(int a, int b, int c) {
    if (a > b) swap(a, b);
    if (a > c) swap(a, c);
    return a;
}

int func(queue<int> que1, queue<int> que2, queue<int> que3) {
    int min = 0x3f3f3f; //初始16位无符号整数的最大值
    while(!que1.empty() && !que2.empty() && !que3.empty())
    {
        int a = que1.front(), b = que2.front(), c = que3.front();
        int min_n = abs(a-b) + abs(a-c) + abs(b-c);
        if(min_n < min) min = min_n;
        int d = min_num(a, b, c);
        if(a == d) que1.pop();
        if(b == d) que2.pop();
        if(c == d) que3.pop();
    }
    return min;
}

int main() {
    int m, n, k, x;
    queue<int> que1, que2, que3;
    cin >> m >> n >> k;
    for (int i = 0; i < m; i++) {
        cin >> x;
        que1.push(x);
    }
    for (int i = 0; i < n; i++) {
        cin >> x;
```

```

        que2.push(x);
    }
    for (int i = 0; i < k; i++) {
        cin >> x;
        que3.push(x);
    }
    cout << func(que1, que2, que3) << endl;
    return 0;
}

```

□□□□□□□□

 alt text □□□□□□


- □□#□□□□□□□□
- □□□□□□□

```

class Solution {
public:
    bool backspaceCompare(string s, string t) {
        stack<char> temp1,temp2;
        for(int i = 0;s[i];i++)
        {
            if(s[i] == '#')    //□□□□□□□□□□
            {
                if(!temp1.empty()) //□□□□□□□ □□□□□□□
                    temp1.pop(); □ □□a##b□□□□#□□□□□□□□□□
            }
            else temp1.push(s[i]);
        }
        for(int i = 0;t[i];i++)
        {
            if(t[i] == '#')
            {
                if(!temp2.empty())
                    temp2.pop();
            }
            else temp2.push(t[i]);
        }
        if(temp1.size() != temp2.size()) return false;
        while(!temp1.empty())
        {
            if(temp1.top() != temp2.top() ) return false;
            temp1.pop();
            temp2.pop();
        }
        return true;
    }
};

```

□□□□

 alt text

- □□□□
- □□□□□□□□□□□□□□
- □□□□
 1. □□□□□□□□a1□a2□a3□a4□□□□□□x
 2. □□□□a2□□□□a2□x□□
 3. a2 > x □□a2□□□□□□□□□□□□□□a2□□
 4. a2 = x □□□□□□□□□□□□x□□□a2□□□
 5. a2 < x □□a2□□□□□□□□□□□□□□□□

- □□□□next_permutation□□□□□□□□

- □□□□□□□□□□□□□□□□

- **next_permutation**□□□□

```
#include #include #include #include #include #include using namespace std; bool valid(int a[],int n) { stack s; int x = 1; for(int i = 0;i < n;i++) { if(s.empty() || s.top() < a[i]) //□□□□□□ □□ □□□□ { while (x<=a[i]) s.push(x),x+=1; } if(s.top() != a[i]) return false; s.pop(); } return true; } int main() { int a[25],n,cnt = 20; cin >> n; for(int i =0;i < n;i++) a[i] = i + 1; do { if(valid(a,n)) { for(int i = 0;i < n;i++) { cout << a[i]; } cout << endl; cnt-=1; } }while(next_permutation(a,a+n) && cnt); return 0; }
```

□□□□□□□□□□

```
#include<iostream>
#include<cstdlib>
#include<cstdio>
#include<string>
#include<cstring>
#include<stack>
#include<algorithm>
#include<queue>
using namespace std;
int vis[25]={0};
int a[25],b[25];
int times = 0;
bool valid(int n)
{
    stack<int> s;
    int x = 1;
    for(int i = 0;i < n;i++)
    {
        if(s.empty() || s.top()<a[i])
        {
            while (x<=a[i])
            {
```

```

        s.push(x);
        x+=1;
    }
}
if(s.top()!=a[i]) return false;
s.pop();
}
return true;
}
void print(int n)
{
    for(int i = 0;i < n;i++)
    {
        cout << b[i];
    }
    cout << endl;
}

void f(int i,int n)
{
    if(times == 20) return ;
    if(i == n && valid(n))
    {
        print(i);
        times ++;
        return;
    }
    for(int j = 1;j <= n;j++)
    {
        if(vis[j]) continue;
        b[i] = j;
        a[i] = j;
        vis[j] = 1;
        f(i+1,n);
        vis[j] = 0;
    }
}

int main()
{
    int n;
    cin >> n;
    f(0,n);
}

```

next_permutation

- `include< algorithm >`
- `bool next_permutation(begin, end)`
 - `begin` 起始位置
 - `end` 结束位置
- 返回 `false` 表示排列已经是最小的排列
- 返回 `true` 表示排列还有下一个排列

- do while
 - while
 -

alt text

-
- pop
-
- return false
 - pop

alt text

```
class Solution {
public:
    bool validateStackSequences(vector<int>& pushed, vector<int>& popped) {
        int x = 0,n = pushed.size();
        stack<int> s;
        for(int i = 0;i < n;i++)
        {
            if(s.empty() || s.top() != popped[i])
            {
                while(x < pushed.size() && pushed[x] != popped[i])
                {
                    s.push(pushed[x]);
                    x += 1;
                }
                if(x == pushed.size()) return false;
                s.push(pushed[x]);
                x +=1 ;
            }
            s.pop();
        }
        return true;
    }
};
```

alt text

```
#include<iostream>
#include<stack>
using namespace std;
stack<int> s;
```

```

char str[10005];
int match[10005];
int main()
{
    cin >> (str + 1); //字符串1000
    for(int i = 1;str[i];i++)
    {
        switch(str[i])
        {
            case '(':
            case '[':
            case '{': s.push(i); break;
            case ')':
                if(!s.empty() && str[s.top()] == '(')
                {
                    match[s.top()] = i; //s.top()是下标
                    s.pop();
                }
                else s.push(i); //是下标
                break;
            case ']':
                if(!s.empty() && str[s.top()] == '[')
                {
                    match[s.top()] = i;
                    s.pop();
                }
                else s.push(i);
                break;
            case '}':
                if(!s.empty() && str[s.top()] == '{')
                {
                    match[s.top()] = i;
                    s.pop();
                }
                else s.push(i);
                break;
        }
    }
    int temp_ans = 0,ans = 0,i = 1;
    while(str[i])
    {
        if(match[i])
        {
            temp_ans += (match[i] - i + 1); //匹配 += 匹配数 * 1
            i = match[i] + 1; //下一个匹配位置
        }
        else
        {
            temp_ans = 0;
            i++;
        }
    }
}


```

```

        if(temp_ans > ans) ans = temp_ans; //更新答案
    }
    cout << ans;
    return 0;
}

```

实验四

 alt text

实验四


- 实验四
- 实验四
- 实验四

```

struct Node { int data; Node *next; }; class MyCircularQueue { public: int count,size; Node
*head,*tail; MyCircularQueue(int k) { head = new Node(); tail = head; for(int i = 0;i < k;i++) {
tail->next = new Node(); tail = tail->next; } count = 0; size = k; tail->next = head; }
bool enqueue(int value) { if(isFull()) return false; tail = tail->next; tail->data = value;
count+=1; return true; }
bool dequeue() { if(isEmpty()) return false; head = head->next; count-=1; return true; }
int Front() { if(isEmpty()) return -1; return head->data; }
int Rear() { if(isEmpty()) return -1; return tail->data; }
bool isEmpty() { return count == 0; } bool isFull() { return count == size; } };

```

实验四

 alt text

- 实验四
 - 实验四
 - 实验四
 - 实验四
- 实验四
 - 实验四100
 - 实验四1000
 - 实验四
 - 实验四0
- 实验四
 - pos
 - cur_pri
 - temp_pri
 - pri-1
 -
 - 1pos

- 7-5-2 7-5

```
#include<bits/stdc++.h>
using namespace std;
#define INF 0x3f3f3f3f
string str;
bool is_operator(char c)
{
    switch(c)
    {
        case '+':
        case '-':
        case '*':
        case '/':
        case '^': return true;
        default : return false;
    }
    return false;
}

long long result(string &s, long long l, long long r)
{
    long long pos = -1 , pri = INF - 1 , cur_pri , temp_pri = 0;
    for(long long i = l; i < r; i++)
    {
        cur_pri = INF;
        switch(s[i])
        {
            case '(':
                temp_pri += 100;
                break;
            case ')':
                temp_pri -= 100;
                break;
            case '+':
            case '-':
                cur_pri = 1 + temp_pri;
                break;
            case '*':
            case '/':
                cur_pri = 2 + temp_pri;
                break;
            case '^':
                cur_pri = 3 + temp_pri;
                break;
        }
        if((s[i] == '-' || s[i] == '+') && (i - 1 < 0 ||
is_operator(s[i - 1])))
            cur_pri += 1000;
        if(pri >= cur_pri) //
        {
            pri = cur_pri; //
        }
    }
}
```

```

        pos = i;
    }
}
if(pos == -1)
{
    long long num = 0;
    for(long long i = l; i < r; i++)
    {
        if(s[i] < '0' || s[i] > '9') continue;
        num = num * 10 + (s[i] - '0');
    }
    return num;
}
else
{
    long long a = result(s, l, pos);
    long long b = result(s, pos + 1, r);
    switch(s[pos])
    {
        case '+': return a + b;
        case '-': return a - b;
        case '*': return a * b;
        case '/': return a / b;
        case '^': return pow(a, b);
    }
}
}
int main()
{
    cin >> str;
    cout << result(str, 0, str.size());
    return 0;
}

```