撞石头

小花正在对一堆石头进行对撞实验：每次从所有石头中选出最重的两块石头进行对撞。对撞处理结果是: 如果两块一样重，则两块石头被完全撞碎；如果两块石头不一样重，那么对撞会剩下一块重量为两者之差的石头放回石头堆中。如此循环对撞。实验结果：如果剩下一块石头，则为此石头的重量；如果没有石头剩下，则为 0。请你设计C/C++程序，根据这堆石头中每块石头的重量计算实验结果。

- 输入格式
  - 第一行，一个正整数n（代表石头的个数）
  - 第二行， n个正整数（代表每个石头的重量，<= 10000）
- 输出格式
  - 一行，一个正整数（代表最后剩下那块石头的重量）

输入：
5
5 2 7 10 10
预期输出：
0

解法一：反复排序

```
int main(){
    int n, w[200], i = 0; cin >> n;
    for (int i = 0; i < n; ++i) cin >> w[i];
    i = 0;
    while (i < n - 1){
        MySort(w, i, n);//排序，从下标为i的元素开始从大到小，之前全0

        if (w[i] == w[i + 1]){
            w[i] = w[i + 1] = 0;
            i += 2;
        }
        else{
            w[i+1] = w[i]-w[i+1], w[i] = 0;
            ++i;
        }
    }
    cout << (i < n ? w[i] : 0) << endl;
    return 0;
}
```

解法一：反复排序

```
void MySort(int *pw, int i, int n)//从下标为i的元素开始从大到小排序
{
    for(int k = i; k < n-1; ++k)
    {
        int max = k;
        for(int j = k+1; j < n; ++j)
            if(pw[max] < pw[j])
                max = j;
        if(max != k)
        {
            int temp = pw[max];
            pw[max] = pw[k];
            pw[k] = temp;
        }
    }
}
```

解法二：不排序，求最大二值、更新

```
int main()
{
    int n, w[200], i = 0;
    cin >> n;
    for (int i = 0; i < n; ++i)
        cin >> w[i];

    ……   //函数1:选出最大的两个石头所处的位置
    ……   //函数2:比较大小之后对石头数组进行更新


    return 0;
}
```

解法二：不排序，求最大二值、更新

```
void Select2Max(int *pw, int n, int *first, int *second)
{
    for (int i = 0; i < n; i++)
    {
        if (*(pw + i) == -1) continue; // 当前位置处的石头已销毁
        if (*first == -1) *first = i;  // first位置若为-1，设置为当前第i处为最大

        // 如果有更大的，更新first和second的位置
        if (*(pw + i) > *(pw + *first)) {
                *second = *first;
                *first = i;
        }
        else {// 更新第二大的石头的位置
                if (*second == -1 && *first != i) *second = i;
                if (*second != -1 && *(pw + i) >= *(pw + *second)) {
                        *second = i;
                }
        }
    }
}
```

解法二：不排序，求最大二值、更新

```
while (1) {
        int first = -1, second = -1;
        Select2Max(w, n, &first, &second);

        if (second == -1 && first != -1) {
                printf("%d\n", *(w + first)); break;
        }
        if (first == -1 && second == -1) {
                printf("0\n"); break;
        }

        if (*(w + first) == *(w + second)) {
                *(w + first) = -1;
                *(w + second) = -1; // 都被撞碎
        }
        else {
                *(w + first) -= *(w + second); // 剩下的石头之差
                *(w + second) = -1; // 另一个被撞碎
        }
}
```

检查圆括号是否配对

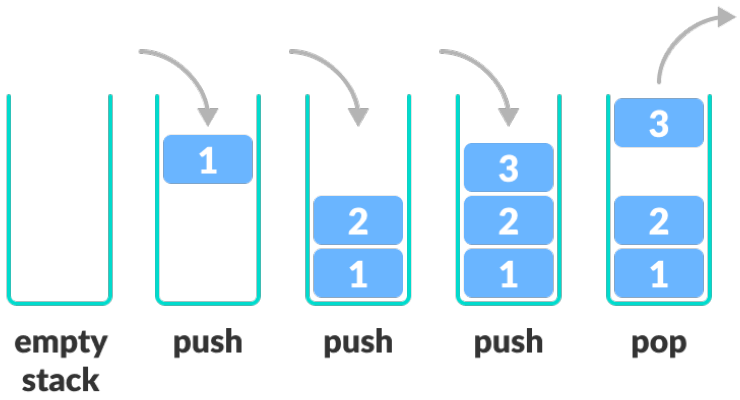
请你设计C/C++程序，对输入的一个算术表达式（字符串），检查其中的圆括号配对情况，如果配对则输出Y，否则输出N。（多左括号、多右括号或左右括号颠倒均算作不配对）。

- 输入格式
  - 第一行，一个正整数n（表达式的长度）
  - 第二行，一个字符串
- 输出格式
  - 一个字母（要么是Y；要么是N）

输入：
15
(x+y))a+b((x-y)
预期输出：
N

解法一：左右括号消消乐（栈的思想）

```
bool expMatch(const char str[])
{     int count = 0;
      for(int i=0; i < strlen(str); ++i)
      {     if (str[i] == '(' )
                  ++count;
            else if (str[i] == ')' )
                  --count;
            if(count < 0)          // 中途count<0，说明先出现 ‘)’
                  return false;
      }
      if (count == 0 )
            return true;
      else
            return false;
}
```



反映“配对/不配对”两种情况

解法一：左右括号消消乐（栈的思想）

```
int expMatch(const char str[])
{     int count = 0; //可能小于0
      for(int i=0; i < strlen(str); ++i)
      {     if (str[i] == '(' )
                  ++count;
            else if (str[i] == ')')
                  --count;
      }
       return count; //>0:多左括号，<0:多右括号，0:一样多
}
```

str[i] != '\0'

只能反映“多左/多右/一样多”三种情况（一样多可能配对，也可能不配对）

虽然可以通过测试用例，但是可能代码是错的！

解法二：搜索的思想

搜索出所有匹配的括号

```
int main() {
    int n;
    scanf("%d", &n);
    getchar();
    char a[100];
    scanf("%s", a, 100);
    offset(a, 0, n);
    int flag = 1;
    for (int i = 0; i < n && flag==1; i++)
        if (a[i] == '(' || a[i] == ')') {
            printf("N"); flag=0;
        }
    if (flag == 1) printf("Y");
    return 0;
}
```

解法二：搜索的思想

```
void offset(char str[], int start, int end) {
      for (int i = start; i < end; i++) {
            if (str[i] == '('
                  for (int j = i + 1; j < end; j++){
                        if (str[j] == ')') {
                              str[j] = '0';
                              str[i] = '0';
                              break;
                        }
                        if (str[j] == '('
                              offset(str, j, end);
                  }
      }
}
```

回文串

请你设计C/C++程序，判别一个字符串是否为回文串（从左到右或者从右到左读起来都一样的字符串）。

- 输入格式（1行）
  - 一个字符串
- 输出格式
  - 一个字母（要么是Y；要么是N）

输入：
level
预期输出：
Y

输入：
895323598
预期输出：
Y

解法一：遍历、比较

```
int main()
{
	char str[100];
	scanf("%s", str);

	int len = strlen(str);
	int i;

	for (i=0; i < len-1; ++i)
		if(str[i] != str[len-i-1]){
			printf("N");
			return 0;
		}

	printf("Y");
	return 0;
}
```

解法一：遍历、比较

```
int main()
{
	char str[100];
	scanf("%s", str);

	int len = strlen(str);
	int i = 0, j = len - 1;

	while (i < j){
		if(str[i] != str[j]){
			printf("N");
			return 0;
		}
		i++; j--;
	}

	printf("Y");
	return 0;
}
```