Educational Codeforces Round 94 (Rated for Div. 2)

A. String Similarity

根据题意，逐个字符构造即可。

```
#include <bits/stdc++.h>

using namespace std;
typedef long long LL;
int main(){
    int t;
    cin>>t;
    while(t--) {
        int n;
        cin >> n;
        string s;
        cin >> s;
        for (int i = 0; i < n; ++i) {
            putchar(s[i*2]);
        }
        puts("");
    }
}
```

B. RPG Protagonist

两个背包，两种有限个物品，输出最多可以取多少个物品。 枚举第一次取的情况，第二次贪心取，先取代价小的。

```
#include <bits/stdc++.h>

using namespace std;
typedef long long LL;

int main() {
    int t;
    cin >> t;
    while (t--) {
        int p, f, cnts, cntw, s, w;
        scanf("%d%d%d%d%d%d", &p, &f, &cnts, &cntw, &s, &w);
        int ans = 0;
        for (int i = 0; i <= cnts; ++i) {
            if (s * i > p)break;
```

```
            int j = min(cntw, (p - s * i) / w);
            int cs = cnts - i;
            int cw = cntw - j;
            int tot = i + j;
            if(s > w){
                j = min(cw, f/w);
                tot += j + min(cs, (f - j * w) / s);
            }

            else{
                j = min(cs, f/s);
                tot += j + min(cw, (f - j * s) / w);
            }
            ans = max(ans, tot);
        }
        cout<<ans<<endl;
    }
}
```

C. Binary String Reconstruction

这道题相当于对s做一个卷积操作后，由结果反推s 原字符串中一个字符可以对应至多两个新字符串字符，对应2个时原字符串字符为1当且仅当对应的2个字符均为1。对应一个时，与新字符串相同。根据此规则生成原字符串后再做检查即可。

```
#include <bits/stdc++.h>

using namespace std;
typedef long long LL;
char ans[200000] = {};

int main() {
    int t;
    cin >> t;
    while (t--) {
        int x;
        string s;
        cin >> s;
        scanf("%d", &x);
        int n = s.length();
        memset(ans, '1', sizeof(char) * n);
        for (int i = 0; i < n; ++i) {
            if (s[i] == '0') {
                if (i - x >= 0) ans[i - x] = '0';
                if (i + x < n) ans[i + x] = '0';
            }
        }
```

```
        bool ok = true;
        for (int i = 0; i < n; ++i) {
            if (i - x < 0  && i + x < n) {
                ans[i] = s[i + x];
            } else if (i + x >= n && i - x >= 0) {
                ans[i] = s[i - x];
            }
        }

        for (int i = 0; i < n; ++i) {
            if (s[i] == '1') {
                bool okk = (i - x >= 0 && ans[i - x] == '1') || (i + x < n
&& ans[i + x] == '1');
                ok &= okk;
            }
        }

        if(ok) {
            for (int j = 0; j < n; ++j) {
                putchar(ans[j]);
            }
            puts("");
        }
        else puts("-1");


    }
}
```

</br>

D. Zigzags

求1≤i<j<k<l≤n，ai=ak 且 aj=al;的四元组(i,j,k,l)的数量。 二维dp，先求出ai = aj且i<j的二元组(i,j)数量。对于每一个满足ai = aj的(i,j),可以找到dp[i-1][j-1]-dp[i-1][i]个符合条件的四元组。 结果最大为sum n2，需要开long long

```
#include <bits/stdc++.h>

using namespace std;
typedef long long LL;
const int maxn = 3e3 + 10;
int a[maxn];
int dp[maxn][maxn];
int main() {
    int t;
    cin >> t;
```

```
    while (t--) {
        int n;
        cin >> n;
        LL ans = 0;
        for (int i = 1; i <= n; i++)
            scanf("%d", a + i);
        for (int i = 0; i <= n; i++)
            for (int j = 0; j <= n; j++)
                dp[i][j] = 0;
        for (int i = 1; i < n; i++)
            for (int j = i + 1; j <= n; j++) {
                if (a[i] == a[j]) {
                    dp[i][j] = 1;
                }
                else dp[i][j] = 0;
            }
        for (int i = 1; i <= n; i++)
            for (int j = 1; j <= n; j++)
                dp[i][j] += dp[i - 1][j] + dp[i][j - 1] - dp[i - 1][j - 1];

        for (int i = 1; i <= n; i++)
            for (int j = i + 1; j <= n; j++)
                if(a[i] == a[j]) {
                    ans += dp[i-1][j-1] - dp[i-1][i];
                }
        cout<<ans<<endl;
    }
    return 0;
}
```

E. Clear the Multiset

对于一段连续的区间，可以发现“ =|= ”的消除形式一定弱于“ === ”，而“ 丄丄丄 ”一定弱于“ ||| ”。 因此最优情况只可能为尽可能多地进行第二种操作直到不连续，或全部进行第一种操作。

在区间最小值处进行分割，比较两种情况，再对子区间深搜得到答案。

```
#include <bits/stdc++.h>
using namespace std;
typedef long long LL;
const int maxn = 5e3+10;
int a[maxn];
int getans(int t, int l, int r){
    if(r < l) return 0;
    int m = INT32_MAX;
    int j = 0;
    for (int i = l; i <= r; ++i) {
        if(a[i] < m)m = a[i], j = i;
```

```
    }
    return min(r - l + 1, m - t + getans(m, l, j-1) + getans(m, j+1, r));

}
int main() {
        int n;
        cin >> n;
        for (int i = 0; i < n; ++i) {
            scanf("%d", a+i);
        }
        int m;
        cout<<getans(0, 0, n-1)<<endl;
    return 0;
}
```