【Brackets Sequence】

```
#include<cstdio>
#include<cstring>
const int N=100;
char str[N];                    //Input String
int dp[N][N];
int path[N][N];
void oprint(int i,int j) //output regular brackets sequence containing subsequence str[i,
j]
{
    if(i>j)
      return;
    if(i==j)                      //there is only one character for subsequence str[i, j]
      {
        if(str[i]=='['||str[i]==']')
          printf("[]");
        else
          printf("()");
      }
    else if(path[i][j]==-1)       // str[i] and str[j] are matched brackets
        {
        printf("%c",str[i]);
        oprint(i+1,j-1);
        printf("%c",str[j]);
        }
       else                       // otherwise
       {
         oprint(i,path[i][j]);
         oprint(path[i][j]+1,j);
      }
}
int main(void)
{
    while(gets(str))
      {
        int n=strlen(str);
        if(n==0)
```

```
                {
                    printf("\n");
                    continue;
                }
            memset(dp,0,sizeof(dp));
            for(int i=0;i<n;i++)
                dp[i][i]=1;
            for(int r=1;r<n;r++)                //Stage: r is the length of subsequences
                {
                    for(int i=0;i<n-r;i++)           //State: fronts of subsequences are enumerat
ed
                        {
                            int j=i+r;                    // rears of subsequences
                            dp[i][j]=0x7fffffff;          // Initialization
                            if((str[i]=='(' && str[j]==')') || (str[i]=='[' && str[j]==']')) // str[i] and str[j]
 are matched
                                {
                                    dp[i][j]=dp[i+1][j-1];
                                    path[i][j]=-1;
                                }
                            for(int k=i; k<j; k++)             // k is enumerated
                                {
                                    if(dp[i][j]>dp[i][k]+dp[k+1][j])
                                        {
                                            dp[i][j]=dp[i][k]+dp[k+1][j];
                                            path[i][j]=k;
                                        }
                                }
                        }
                }
            oprint(0,n-1);                  // Output the regular brackets sequence
            printf("\n");
        }
    return 0;
}
```

【Dollars】

```
#include <iomanip>
#include <iostream>
using namespace std;
int main(void) {
    int b[] = {1, 2, 4, 10, 20, 40, 100, 200, 400, 1000, 2000}; //5c coin is used as the unit for notes and coins for New Zealand currency
    long long a[6001] = {1}; // the number of ways in which n 5c coins may be made up using notes and coins for New Zealand currency is a[n]
    //Off-line method, DP
    for (int i = 0; i < 11; i++){          // Enumerate all coins and notes
        for (int j = b[i]; j < 6001; j++) {   // Enumerate
            a[j] += a[j - b[i]];
        }
    }
    cout << fixed << showpoint << setprecision(2);
    for (float fIn; cin >> fIn && fIn != 0; cout << endl) {
        cout << setw(6) << fIn << setw(17) << a[(int)(fIn * 20 + 0.5f)];
    }
    return 0;
}
```

【Longest Match 】

```
#include<iostream>
#include<cstring>
#include<cstdio>
#include<string>
#include<algorithm>
#define N (1024)
using namespace std;
struct text{                    // two successive lines of string
    int num;                    // number of words
    string word[1024];          // words
}t1,t2;
string s1,s2;
int f[N][N];                    //the number of matched words for the first i-th words in s1a
nd the first j-th words in s2 is f[i, j]
void devide(string s,text &t)// sequence of words t.word[] whose length is t.num is ta
ken out from s
{
    int l=s.size();                 //the length of s
    t.num=1;
    for(int i=0;i<1000;i++) t.word[i].clear();
    for (int i=0;i<l;++i)
        if ('A'<=s[i] && s[i]<='Z' || 'a'<=s[i] && s[i]<='z'||'0'<=s[i]&&s[i]<='9')
            t.word[t.num]+=s[i];
        else  ++t.num;
    int now=0;
    for(int i=1;i<=t.num;i++)   if(!t.word[i].empty())    t.word[++now]=t.word[i];
    t.num=now;
}
int main(void)
{
    int test=0;                     //Initialization: the number of test case
    while (!cin.eof())
    {
        ++test;
        getline(cin,s1);                // Input string s1
        devide(s1,t1);
        getline(cin,s2);                    //Input string s2
        devide(s2,t2);
        printf("%2d. ",test);
        if(s1.empty() || s2.empty())
        {
            printf("Blank!\n");
            continue;
        }
        memset(f,0,sizeof(f));
        for (int i=1;i<=t1.num;++i)   // words in s1
            for (int j=1;j<=t2.num;++j) //words in s2
            {                                   //Calculation
                f[i][j]=max(f[i-1][j],f[i][j-1]);
                if (t1.word[i]==t2.word[j])
                    f[i][j]=max(f[i][j],f[i-1][j-1]+1);
            }
        printf("Length of longest match: %d\n",f[t1.num][t2.num]); // Output result
    }
    return 0;
}
```

【History Grading】

```
#include<iostream>
#include<cstring>
#include<cstdio>
using namespace std;
int n;                              //number of events
int f[30][30];
int st[30];                         // st[t] is the t-th event in the chronological order
int ed[30];                 // ed[t] is the t-th event in the current student's chronological order
int tmp[30];
int main(void)
{
    freopen("111.in","r",stdin);
    freopen("HG.out","w",stdout);
    scanf("%d",&n);                 // Input number of events
    for(int i=1;i<=n;++i)           // Input the correct chronological order of n events
    {
        cin >> tmp[i];
        st[tmp[i]]=i;
    }
    while(!cin.eof())               //Input students' chronological ordering of the n events
    {
        for(int i=1;i<=n;++i)       // Input current student's chronological ordering of the n
events
        {
            cin >> tmp[i];
            ed[tmp[i]]=i;
        }
        if(cin.eof()) break;
        memset(f,0,sizeof(f));
        for(int i=1;i<=n;++i)           //Calculate the LCS for st[ ] and ed[ ]
            for(int j=1;j<=n;++j)
            {
                f[i][j]=max(f[i-1][j],f[i][j-1]);
                if(st[i]==ed[j])
                    f[i][j]=max(f[i][j],f[i-1][j-1]+1);
            }
        cout << f[n][n] << endl;        //Output the current student's score
    }
    return 0;
}
```