HW2_Longest palindrome subsequence

程式說明

本程式會打印給定字串的最長回文子序列。用戶鍵盤輸入將要計算字串的個數，隨後依次輸入字串（字串由字符字母或者數字組成，給定字串長度不會超過 1000)。程式會打印最長回文子序列的長度和序列。

采用的方法是動態規劃，判斷字串是否對稱，減少計算消耗。不斷計算左右元素是不是一樣來達到最終目的。

代碼及注釋在下一頁。

```cpp
#include <iostream>
#include <string.h>
#define MAX 1001   //設置輸入string最大值
using namespace std;


//聲明變量
char str[MAX] = "";
//存儲數據
int store_ans[1000][1000];
int store_ins[1000][1000];

//使用動態規劃尋找最長回文子序列
int find(int i, int j)
{
    if (i == j) return 1;
    if (i > j) return 0;
    if (store_ans[i][j] != -1) return store_ans[i][j];
    if (str[i] == str[j])
        store_ans[i][j] = find(i+1, j-1) + 2, store_ins[i][j] = 0;
    else if (find(i+1, j) > find(i, j-1))
        store_ans[i][j] = find(i+1, j), store_ins[i][j] = 1;
    else if (find(i+1, j) < find(i, j-1))
        store_ans[i][j] = find(i, j-1), store_ins[i][j] = 2;
    else
        store_ans[i][j] = find(i, j-1), store_ins[i][j] = 3;
    return store_ans[i][j];
}

 //打印
void PrintLPS(int i, int j)
{
    if (i > j) return;
    if (i == j)
        cout << str[i];
    else if (store_ins[i][j] == 0)
        cout << str[i], PrintLPS(i+1, j-1), cout << str[i];
    else if (store_ins[i][j] == 1)
        PrintLPS(i+1, j);
    else
        PrintLPS(i, j-1);
}
```

```
 //匯總
void LPS()
{
    memset(store_ans, -1, sizeof(store_ans));
    int N = strlen(str);
    printf("%d\n",find(0, N-1));
    PrintLPS(0, N-1);
    printf("\n");
}
//主函數
int main(){
    int i = 1;
    int times = 0; //測資次數
    scanf("%d", &times);
    while(i <= times){ //循環獲取用戶輸入的字串計算LPS
    scanf("%s",str);
    LPS();
    i++;
    }
}
```