7-2 符号配对 (20 分)

请编写程序检查C语言源程序中下列符号是否配对：/*与*/、(与)、[与]、{与}。

输入格式:

输入为一个C语言源程序。当读到某一行中只有一个句点.和一个回车的时候，标志着输入结束。程序中需要检查配对的符号不超过100个。

输出格式:

首先，如果所有符号配对正确，则在第一行中输出YES，否则输出NO。然后在第二行中指出第一个不配对的符号：如果缺少左符号，则输出?-右符号；如果缺少右符号，则输出左符号-?。

输入样例1：

```
void test()
{
    int i, A[10];
    for (i=0; i<10; i++) /*/
        A[i] = i;
}
.
```

输出样例1：

```
NO
/*-?
```

输入样例2：

```
void test()
{
    int i, A[10];
    for (i=0; i<10; i++) /**/
        A[i] = i;
}]
.
```

输出样例2：

```
NO
?-]
```

输入样例3：

```
void test()
{
    int i
    double A[10];
    for (i=0; i<10; i++) /**/
        A[i] = 0.1*i;
}
.
```

输出样例3：

```
YES
```

```
#include<stdio.h>
#define maxsize 1003
#include<stdlib.h>
typedef struct node *stack;
struct node{
	char ch[maxsize];
	int top;
};
stack creat()
{
	stack s;
	s=(stack)malloc(sizeof(struct node));
	s->top=-1;
	return s;
}
void push(stack s,char cht)
{
	s->top++;
	s->ch[s->top]=cht;
}

int main()
{
	int i,k=0,j;
	static char ch1[1000],ch2[1000],ch[10000];
	struct node *s1=NULL;
	s1=creat();
	ch1['(']=')';
	ch1['{']='}';
	ch1['[']=']';
	//'*/'用'>'来代替;
	//'/*'用'<'来代替;
	ch1['<']='>';
	for(i=0;;i++)
	{
		gets(ch);
		if(ch[0]=='.'&&ch[1]=='\0') break;
		for(j=0;ch[j]!='\0';j++)
		{
			if(ch[j]=='('||ch[j]==')'||ch[j]=='['||ch[j]==']'||ch[j]=='{'||ch[j]=='}')
			{
				ch2[k++]=ch[j];
			}
			else if(ch[j]=='/'&&ch[j+1]=='*')
			{
```

```
                ch2[k++]='<';
                j++;
            }
            else if(ch[j]=='*'&&ch[j+1]=='/')
            {
                ch2[k++]='>';
                j++;
            }
        }
}
int flag=1;
for(i=0;i<k;i++)
{
        if(ch2[i]=='('||ch2[i]=='['||ch2[i]=='{'||ch2[i]=='<')
        {
            push(s1,ch2[i]);
        }
        else if(ch2[i]==')'||ch2[i]==']'||ch2[i]=='}'||ch2[i]=='>')
        {
            if(s1->top!=-1&&ch1[s1->ch[s1->top]]==ch2[i])
            {
                s1->top--;
            }
            else
            {
                printf("NO\n");
                //缺左边的符号;
                if(s1->top==-1)
                {
                    if(ch2[i]==')') printf("?-)");
                    else if(ch2[i]=='}') printf("?-}");
                    else if(ch2[i]==']') printf("?-]");
                    else if(ch2[i]=='>') printf("?-*/");
                }
                //缺右边的符号;
                else if(ch1[s1->ch[s1->top]]!=ch2[i])
                {
                    if(s1->ch[s1->top]=='(') printf("(-?");
                    else if(s1->ch[s1->top]=='[') printf("[-?");
                    else if(s1->ch[s1->top]=='{') printf("{-?");
                    else if(s1->ch[s1->top]=='<') printf("/*-?");
                }
                flag=0;
                break;
            }
        }
}
```

```
    //切记，当输出YES时，堆栈一定为空；
        if(flag==1&&s1->top==-1) printf("YES");
    //当输入字符串为"[[[]"时;
    //左边有多余左符号；
    else if(flag==1&&s1->top!=-1)
    {
        printf("NO\n");
        if(s1->ch[s1->top]=='(') printf("(-?");
              else if(s1->ch[s1->top]=='[') printf("[-?");
                                  else if(s1->ch[s1->top]=='{') printf("{-?");
                                  else if(s1->ch[s1->top]=='<') printf("/*-?");
    }
        return 0;
}
```