7-1 【模板】KMP字符串匹配 (20 分)

给出两个字符串text和pattern，其中pattern为text的子串，求出pattern在text中所有出现的位置。

为了减少骗分的情况，接下来还要输出子串的前缀数组next。

输入格式:

第一行为一个字符串，即为text。

第二行为一个字符串，即为pattern。

输出格式:

若干行，每行包含一个整数，表示pattern在text中出现的位置。

接下来1行，包括length(pattern)个整数，表示前缀数组next[i]的值，数据间以一个空格分隔，行尾无多余空格。

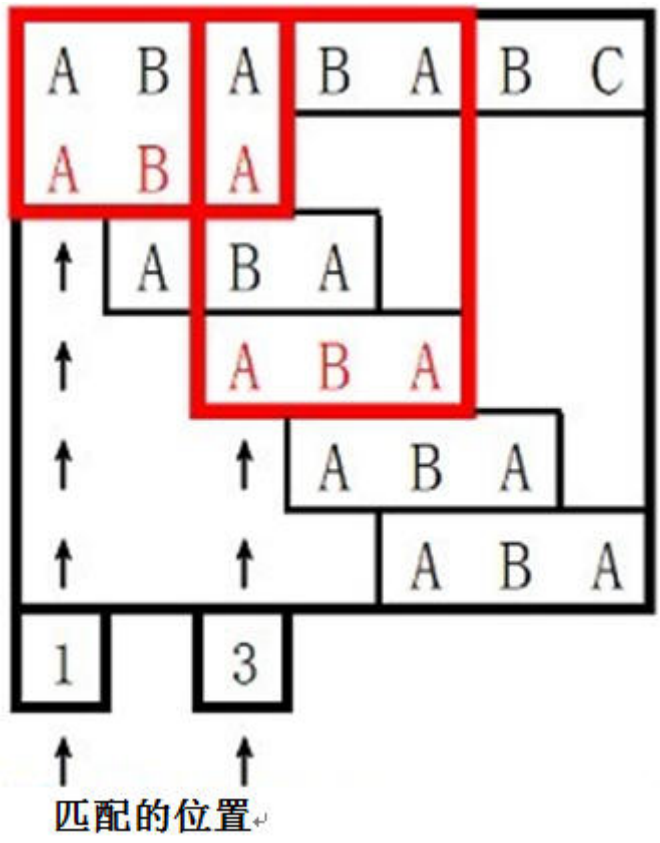
输入样例:

```
ABABABC
ABA
```

输出样例:

```
1
3
0 0 1
```

样例说明:




```
#include<stdlib.h>
#include<stdio.h>
#include<string.h>
```

```
int next[20];
int input(){
	char *text,*pattern;
	text=(char*)malloc(sizeof(char)*20);
	pattern=(char*)malloc(sizeof(char)*20);
	gets(text);
	gets(pattern);
	int temp=0;
	int len=strlen(pattern);
	while(1){
		temp=patter(text,pattern,temp);
		if(temp!=-1){
			printf("%d\n",temp);
			temp=temp+1;
		}else{
			break;
		}
	}
}

int getnext(char *t){
	int i=0;
	int j=-1;
	int len=strlen(t);
	next[0]=-1;
	while(i<len-1){
		if(j==-1||t[i]==t[j]){
			i++;
			j++;
			next[i]=j;
		}else{
			j=next[j];
		}
	}
}

int patter(char *text,char *pattern,int post){
	int tlen=strlen(text);
	int plen=strlen(pattern);
	int i=post;
	int j=0;
	getnext(pattern);

	while(i<tlen&&j<plen){
		if(j==-1||text[i]==pattern[j]){
			i++;
```

```
            j++;
        }else{
            j=next[j];
        }

    }
    if(j==plen){
        return (i-j)+1;
    }else{
        return -1;
    }
}


int main(){
input();
}
```