写一段代码，判断一个包括'{','[','(',')',']','}'的表达式是否合法(注意看样例的合法规则。)
给定一个表达式 **A**,请返回一个 bool 值，代表它是否合法。
测试样例：

```
"[a+b*(5-4)]*{x+b+b*({1+2)}}"
```

返回：true

测试样例：

```
"[a+b*(5-4)]*{x+b+b*(({1+2)}}"
```

返回：false

```
1    import java.util.*;
2
3    public class ChkExpression {
4        public static boolean chkLegal(String A) {
5                StringBuffer sb = new StringBuffer(A);
6                int length = sb.length();
7                Stack<Character> s = new Stack<Character>();
8                for (int i = 0; i < length; i++) {
9                        char c = sb.charAt(i);
10                       if (c == '{' || c == '[' || c == '(' ){
11                               s.push(c);
12                       }
13                       if ( c == '}' || c == ']' || c == ')'){
```

```
14                              s.pop();
15                      }
16              }
17              if(s.isEmpty())
18                      return true;
19              return false;
20      }
21  }
```

从小明家所在公交站出发有 n 路公交到公司，现给出每路公交的停站数(不包括起点和终点)，及每次停的时间(一路车在每个站停的时间相同)和发车的间隔，先假定每辆车同时在相对时间 0 分开始发车，且所有车在相邻两个站之间的耗时相同,都为 5 分钟。给定小明起床的相对时间(相对 0 的分钟数)，请计算他最早到达公司的相对时间。
给定每路车的停站数 **stops**,停站时间 **period**,发车间隔 **interval** 及公交路数 **n**，出发时间 **s**。请返回最早到达时间。保证公交路数小于等于 500，停站数小于等于 50。

```
1   class TakeBuses {
2   public:
3       int chooseLine(vector<int> stops, vector<int> period, vector<int> interval, int n, int s) {
4               // write code here
5               int min=99999;
6               int i,temp;
7               for(i=0;i<n;i++){
8                       temp=s;
9                       if(temp%interval[i]!=0) temp=(temp/interval[i]+1)*interval[i]; //需要等待
10                      temp=temp+stops[i]*(5+period[i]);
11                      if(temp<min) min=temp;
```

```
12                  }
13                  return min+5;
14          }
15  };
```

请你实现一个简单的字符串替换函数。原串中需要替换的占位符为"%s",请按照参数列表的顺序一一替换占位符。若参数列表的字符数大于占位符个数。则将剩下的参数字符添加到字符串的结尾。
给定一个字符串 **A**，同时给定它的长度 **n** 及参数字符数组 **arg** 和它的大小 **m**，请返回替换后的字符串。保证参数个数大于等于占位符个数。保证原串由大小写英文字母组成，同时长度小于等于 500。
测试样例：

```
"A%sC%sE",7,['B','D','F']
```

返回："ABCDEF"

第一种方法的思路是 通过字符串的indexOf 方法和replaceFirst 方法 进行替换字符串中的%s,然后进行判断参数 args 里面的参数是否使用完毕,如果没有使用完毕则进行追加到字符最后。 这种方法的缺点是会在内存中生成很多的冗余字符串。优点是 像我这种菜鸟很容易想到，也更与理解。

```
1   public String formatString(String A, int n, char[] arg, int m) {
2                   int i = 0;
3                    while (A.indexOf("%s") >= 0) {
4                           A = A.replaceFirst("%s", String.valueOf(arg[i]));
5                           i++;
6                    }
7                    while (i < m) {
8                           A += arg[i];
```

```
9                       i++;
10                  }
11                  return A;
12  }
```

第二种方法的思路通过字符串分割 split 方法,把字符串中不是%s 的分割出来，存储到数组中，然后遍历这个数组，进行追加参数,但是有一点需要注意字符的开始字符为%s，这样再数组的第一个位置为空串,进行特殊判断即可。这种方法的有点是有效的利用了内存,减少了很多冗余的字符生成。

```
1   public String formatString(String A, int n, char[] arg, int m) {
2                   // write code here
3                   String[] AArr = A.split("%s");
4                   StringBuffer buffer = new StringBuffer();
5                   int i =  0;
6                   for (String str : AArr) {
7                       if (str.length() == 0) {
8                           buffer.append(arg[i]);
9                       } else {
10                          buffer.append(str + arg[i]);
11                      }
12                      i++;
13                  }
14                  while (i < m) {
15                      buffer.append(arg[i]);
16                      i++;
17                  }
18                  return buffer.toString();
19          }
```

第三种就是通过和正则表达式进行组合,通过 matcher 添加到 buffer 里面，进行替换%s，最后需要注意把最尾部的字符添加到 buffer 里面。

```
1   public String formatString(String A, int n, char[] arg,  int
2               m) {
3                  // write code here
4              Pattern pattern = Pattern.compile("%s");
5   Matcher matcher = pattern.matcher(A);
6   StringBuffer buffer = new StringBuffer();
7   int i =  0;
8   boolean result = matcher.find();
9   while (result)
10  { matcher.appendReplacement(buffer,
11              String.valueOf(arg[i]));
12  result=matcher.find();
13  i++;
14  }
15  matcher.appendTail(buffer);
16  while (i < m)
17  { buffer.append(arg[i]);
18  i++;
19  }
20  return buffer.toString();
21          }
```

现在有一个字符串列表，和一个关键词列表，请设计一个高效算法，检测出含关键字列表中关键字(一个或多个)的字符串。

给定字符串数组 **A** 及它的大小 **n** 以及关键词数组 **key** 及它的大小 **m**，请返回一个排好序的含关键词的字符串序号的列表。保证所有字符串长度小于等于100，关键词个数小于等于 100，字符串个数小于等于 200。保证所有字符串全部由小写英文字符组成。若不存在含关键字的字符串，请返回一个只含-1的数组。

测试样例：

```
["nowcoder","hello","now"],3,["coder",now],2
```

返回：[0,2]

```
importjava.util.*;
    publicclassKeywordDetect {

        publicint[] containKeyword(String[] A, intn, String[] keys, intm) {

                // write code here

                intnum=0;

                  List<Integer> list=newArrayList<Integer>();

                  for(inti=0;i<n;i++){

                        for(intj=0;j<m;j++){

                              if(A[i].contains(keys[j])){ lis
```

```
18
19                                  num++;
20
21                                  break;//如果字符串列表含有关键字就跳出结束循环，继续下次循环
22
23                              }
24
25                          }
26
27                  }
28
29                  int[] res=newint[num];
30
31                  if(num==0)
32
33                          returnnewint[]{-1};
34
35                  for(intj=0;j<list.size();j++){
36
37                          res[j]=list.get(j);
38
39                  }
40
41              returnres;
42
43      }
```

```
44
45  }
```