观察下`abb`这个字符串,将其反转得到`bba`,如果只是想得到回文串,那把这个反转的字符串放在前面得到`bbaabb`就肯定是了,但是并非添加最少的字符。

将该反转字符串放在后面观察一下:`abbbba`,前缀和后缀的情况有这些:

```
a                   a
ab                  ba
abb                 bba
abbb                 bbba
```

显然,这个合并的字符串长度最长的相同前后缀是a,这时候把反转后的字符串`bba`中最后的a去掉,得到bb, 再把`bb`接到原字符串前面,得到`bbabb`,这就是最短的回文拼接方法了！

再用一个例子,比如`aaba`,翻转得到`abaa`,然后拼接起来得到`aabaabaa`, 其最长公共前后缀是`aa`,去掉这个后缀的反转字符串是`ab`,在接到原字符串上就是`abaaba`,即最短的回文拼接。

那如何求这个最长公共前后缀有多长那？

```
a a b a a b a a
0 1 0 1 2 3 1 2
```

第一个a是0,因为没有区分前后缀

第二个a是1,因为在第2个位置,可以有最长为1的相同前后缀`a` 依次类推,只要知道最后一个字母对应的数,就是这个字符串的最长公共前后缀长度了,如何求next[]那？

pos表示前缀匹配到的位置,index表示后缀匹配到的位置

```
vector<int> getNext(string str){
    int len=str.size();
    vector<int>next(len,0);
    int index=1,pos=0;
    while(index<len){
        while(pos>0 && str[pos]!=str[index] ){
            pos=next[pos-1];
        }
        if(str[pos]==str[index]){
            pos++;
        }
        next[index]=pos;
        index++;
    }
    return next;
}
```

如果相等,pos向后移动

否则移动到next[pos-1]的位置

同时为了方便处理空字符串,在反转拼接的时候加了`#`,这个字符要保证不会出现在字符串中。