有一个长为 n 的数组 A，求满足 0≤a≤b<n 的 A[b]-A[a]的最大值。
给定数组 **A** 及它的大小 **n**，请返回最大差值。
测试样例：

```
[10,5],2
```

返回：0

```
1   importjava.util.*;
2
3   public class LongestDistance {
4           public int getDis(int[] A,  int n)
5                   int dis=0;
6                   if(n>1){
7                           int min=A[0];
8                           for(int i=1;i<n;i++){
9                                   if(A[i]-min>dis){
10                                          dis=A[i]-min;
11                                  }
12                                  if(min>A[i]){
13                                          min=A[i];
14                                  }
15                          }
16                  }
17                  return dis;
18          }
```

```
19    }
```

在 4x4 的棋盘上摆满了黑白棋子，黑白两色的位置和数目随机其中左上角坐标为(1,1),右下角坐标为(4,4),现在依次有一些翻转操作，要对一些给定支点坐标为中心的上下左右四个棋子的颜色进行翻转，请计算出翻转后的棋盘颜色。
给定两个数组 **A** 和 **f**,分别为初始棋盘和翻转位置。其中翻转位置共有 3 个。请返回翻转后的棋盘。
测试样例：

```
[[0,0,1,1],[1,0,1,0],[0,1,1,0],[0,0,1,0]],[[2,2],[3,3],[4,4]]
```

返回：`[[0,1,1,1],[0,0,1,0],[0,1,1,0],[0,0,1,0]]`

```
1    public static int[][] flipChess(int[][] A,  int[][] f) {
2                    // write code here
3                    for (int i =  0; i < f.length; i++) {
4                            int row = f[i][0] -  1;
5                            int col = f[i][1] -  1;
6
7                            if (row -  1 >=  0) {
8                                    A[row -  1][col] = (A[row -  1][col] ==  0) ?  1 :  0;
9                            }
10
11                           if (row +  1 <=  3) {
12                                   A[row +  1][col] = (A[row +  1][col]) ==  0 ?  1 :  0;
13                           }
14
15                           if (col -  1 >=  0) {
```

```
16                              A[row][col - 1] = (A[row][col - 1]) == 0 ? 1 : 0;
17                      }
18
19                      if (col + 1 <= 3) {
20                              A[row][col + 1] = (A[row][col + 1]) == 0 ? 1 : 0;
21                      }
22              }
23              return A;
24      }
```

现在有一个城市销售经理，需要从公司出发，去拜访市内的商家，已知他的位置以及商家的位置，但是由于城市道路交通的原因，他只能在左右中选择一个方向，在上下中选择一个方向，现在问他有多少种方案到达商家地址。
给定一个地图 **map** 及它的长宽 **n** 和 **m**，其中 1 代表经理位置，2 代表商家位置，-1 代表不能经过的地区，0 代表可以经过的地区，请返回方案数，保证一定存在合法路径。保证矩阵的长宽都小于等于 10。
测试样例：

```
[[0,1,0],[2,0,0]],2,3
```

返回：2

```
1   import java.util.*;
2
3   public class Visit {
4           public int countPath(int[][] map,  int n,  int m) {
5                   // write code here
6                   int x1 = -1,y1 = -1;//经理的坐标
```

```
7                 int x2 = -1,y2 = -1;//商家的坐标
8                 for(int i =  0;i<n;i++){
9                         for(int j =  0; j<m;j++){
10                                if(map[i][j]==1){
11                                        x1 = j;
12                                        y1 = i;
13                                }else if(map[i][j]==2){
14                                        x2 = j;
15                                        y2 = i;
16                                }
17                        }
18                }
19                int xto = x1>x2?-1:1;//根据经理和商家的方向判断向左还是向右走
20                int yto = y1>y2?-1:1;//向上还是向下
21                //动态规划的思想 map[y][x]记录着经理到 x,y 点最多的路程数
22                for(int y = y1;y!=(y2+yto);y+=yto){
23                        for(int x = x1;x!=(x2+xto);x+=xto){
24                                if(y==y1||x==x1){
25                                        map[y][x] =  1;
26                                        continue;
27                                }
28                                map[y][x] = map[y-yto][x]+map[y][x-xto];
29                        }
30                }
31                return map[y2][x2];
32        }
```

```
33  }
```

有一个直方图，用一个整数数组表示，其中每列的宽度为 1，求所给直方图包含的最大矩形面积。比如，对于直方图[2,7,9,4],它所包含的最大矩形的面积为 14(即[7,9]包涵的 7x2 的矩形)。
给定一个直方图 **A** 及它的总宽度 **n**，请返回最大矩形面积。保证直方图宽度小于等于 500。保证结果在 int 范围内。
测试样例：

```
[2,7,9,4,1],5
```

返回：14

```
1   int i,j,L1,L2;
2           int max=0;
3           for(i=0;i<n;i++)
4           {
5                   L1=0;L2=0;
6                   for(j=i;j<n;j++)
7                   {
8                           if(A[j]>=A[i])L1++;
9                           else break;
10
11                  }
12                  for(j=i-1;j>=0;--j)
13                  {
14                          if(A[j]>=A[i])L2++;
15                          else break;
```

```
16                  }
17                  area=(L1+L2)*A[i];
18                  if(area>max)max=area;
19          }
20          //printf("max area:%d\n",max);
21          return max;
```

求字典序在 s1 和 s2 之间的，长度在 len1 到 len2 的字符串的个数，结果 mod 1000007。

```
1   #include<iostream>
2   #include<string>
3   #include<vector>
4   #include<math.h>
5   using namespace std;
6
7   int main(){
8           //根据题中给出的例子，这个字符串只包含小写字母，不然答案就不应该是 56 了
9           string s1,s2;
10          int len1,len2;
11          while(cin>>s1>>s2>>len1>>len2){
12                  //只包含小写字母的字符串可以看成 26 进制的数制
13                  //将 s1 和 s2 补长到 len2 长度
14                  s1.append(len2-s1.size(),'a');
15                  s2.append(len2-s2.size(),(char)('z'+1));
16                  vector<int> array;
17                  for(int i=0;i<len2;i++){
```

```
18                          array.push_back(s2[i]-s1[i]);
19                  }
20                  int result = 0;
21                  for(int i=len1;i<=len2;i++){
22                          for(int k=0;k<i;k++){
23                                  result += array[k]*pow(26,i-1-k);
24                          }
25                  }
26                  //所有字符串最后都不包含是 s2 自身，所以最后要减 1；
27                  cout<<result-1<<endl;
28          }
29          return 0;
30  }
```

已知某公司总人数为 W，平均年龄为 Y 岁(每年 3 月末计算，同时每年 3 月初入职新人)，假设每年离职率为 x，x>0&&x<1,每年保持所有员工总数不变进行招聘，新员工平均年龄 21 岁。
从今年 3 月末开始，请实现一个算法，可以计算出第 N 年后公司员工的平均年龄。(结果向上取整)。

```
1   #include <cstdio>
2   #include <string>
3   #include <cstdlib>
4   #include <cmath>
5   #include <vector>
6   #include <iostream>
7   #include <algorithm>
8   #include <queue>
```

```
9
10  using namespace std;
11
12  int w, n;
13  double y, x;
14
15  int main()
16  {
17          freopen("in.txt",  "r", stdin);
18          while (scanf("%d%lf%lf%d", &w, &y, &x, &n) != EOF)
19          {
20                  double newPerson = w * x;//这里必须用 double
21
22                  double aveAge = y;
23                  for (int i = 0; i < n; i++)
24                  {
25                          // 老员工 多了一岁， 新员工总是 21
26                          double totalAge = (w - newPerson) * (aveAge + 1) + newPerson * 21;
27                          aveAge = totalAge / w;
28                  }
29                  int age =  ceil(aveAge);  // 向上取整
30                  printf("%d\n", age);
31          }
32
33          return 0;
34  }
```