HW2-1 Book Donation Program

Description

Twilight Sparkle is an aspiring pony lives in Equestria. She has a large collection of books, all of which are stored in bookcases in her house. For each bookcase, the books stored in it is labelled 1, 2, 3, ... and so on.

One day, she decides to participate in a book donation program. For each bookcase in her house, she wants to pick up one book for donation.

Here is how Twilight chooses the books :

Each time she will ask her friend Spike to give her **two random number x and n**. Twilight wants to find out the **n-th smallest prime p that is equal or greater than x**. For this bookcase, Twilight will then pick up the book labelled **p** for donation.

Since Twilight is not good at mathematics, she asks you to write a program to calculate the answer of all bookcases. If the program is correct, Twilight will be happy and she may teach you some magic tricks.

Input

First line contains one integer **T**, representing the number of the bookcases.

The next T lines contains two integers **x**, **n**, representing the two numbers Spike gives to Twilight.

1 <= T <= 100

1 <= x <= 105

1 <= n <= 1000

Output

For each bookcase, please output a line contains one integer **p**, representing the label of the book that is donated.

Sample Input

```
3
2 1
2 2
10 2
```

Sample Output

```
2
3
13
```

```
#include<stdio.h>

//Find smallest prime number greater than given nth
int t, x, nth;
int main()
{
    scanf("%d", &t);
    //check t times
    for(int i=0; i<t; i++)
    {
        scanf("%d %d", &x, &nth);
        printprime();
    }
    return 0;
}

//print needed prime number
int printprime()
{
    int pn = 0;
    for(int i=x; i<100000; i++)
    {
        if (isPrime(i) == 1)
        {
            pn++;
            if (pn == nth)
            {
                printf("%d\n", i);
                break;
            }
        }
    }
    return 0;
}

// This is the faster version of prime checking function
int isPrime(int n)
{
    if (n == 1) return 0;
    if (n%2==0 && n!=2) return 0;
    int i=0;
    for (i=3; i*i<=n; i+=2) if (n%i==0) return 0;

    return 1;
}
```

HW2-2 Change the Cap

Description

Johnson, a pro table tennis player, is a member of school team in NTHU. The coach trains the whole team three times a week. After training, Johnson needs to drink a lot of sports drinks named "Fin & Bubblegum" (also known as "F&B").
One day, the company decides to hold a long-term promotion. After you collect three caps from the bottle, you can return them back to the company and get a new bottle of "F&B".
Johnson notices that he needs to drink **N** bottle of "F&B" every week. He would like to know the minimum number of "F&B" he needs to buy every week.
If you can solve this problem, maybe Johnson will teach you how to play table tennis.
For this problem, **you don't need to calculate the remaining caps from last week.**
Attention : Johnson found out that some programs may give him wrong answer in some other cases, so he decided to add more testcases to ensure the correctness of your program. (rejudge in 10/5 18:20)

Input

There is only one integer for each testcase.
1 <= N <= 104

Output

Please output the minimum number of "F&B" Johnson needs to buy every week.
Remember to print '\n' after your answer.

Sample Input

13

Sample Output

9

```
#include<stdio.h>

int drink, buy = 0;
int main()
{
    scanf("%d", &drink);
    for(int i=0;i<drink;i++)
    {
        if(i == 0)
        {
            buy = buy + 1;
        }
        else if ((i-3)%3 == 0)
        {
            buy = buy;
        }
        else
        {
            buy = buy + 1;
        }
    }
    printf("%d\n", buy);
}
```

HW2-3 Mexican Wave

Description

James, a pro photographer, recently went to a baseball game. He was very interested in Mexican wave, so he decided to take some photos of it.
Image that there are **n** spectators in the stadium, labelled from **1 to n**. The maximum length of the wave is **m**. The spectators start the Mexican wave at time 0.

- At time 1, the first spectator stands.
- At time 2, the second spectator stands.
- ...
- At time m, the m-th spectator stands.
- At time m + 1, the (m + 1)-th spectator stands and the first spectator sits.
- At time m + 2, the (m + 2)-th spectator stands and the second spectator sits.
- ...
- At time n, the n-th spectator stands and the (n - m)-th spectator sits.
- At time n + 1, the (n + 1 - m)-th spectator sits.
- ...
- At time n + m, the n-th spectator sits.

Now, James prints out **T** pictures he took and wants to play a game with you. For each picture, James will give you **n**, **m**, and the time **t** he took the picture. Can you predict how will picture looks like?
If you win the game, maybe James will teach you some tips about shooting an astonishing picture.

Input

The first line contains one integer number **T**, representing the numbers of photos.
The next T lines contain three integers **n**, **m**, **t**, representing the number of spectators, the maximum length of the wave, and the time James took the picture.
1 <= T <= 100
1 <= m <= n <= 100
1 <= t <= 100

Output

For each picture please output a line contains **n** characters, representing the states of the n spectators. If a spectator is sitting, please print **'-'**, otherwise please print **'^'**. You can find out more information in sample output.

Sample Input

```
10
5 3 0
5 3 1
5 3 2
5 3 3
5 3 4
5 3 5
5 3 6
5 3 7
5 3 8
5 3 9
```

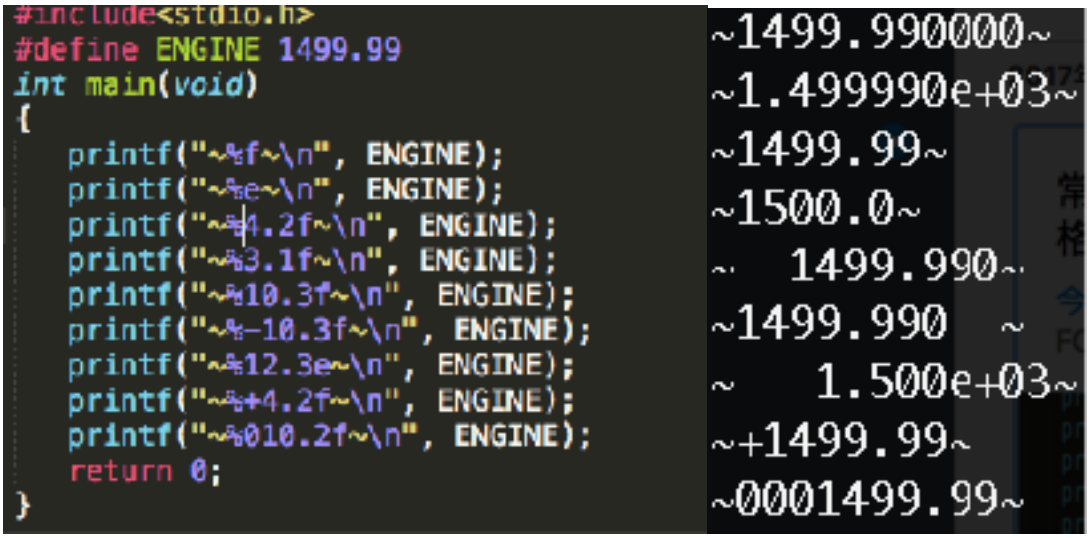
Sample Output

```
-----
^----
^^---
^^^--
-^^^-
--^^^
---^^
----^
-----
-----
```

```
#include <stdio.h>

int n, m, t;
int main()
{
    int T;
    scanf("%d", &T);
    for(int i=0;i<T;i++)
    {
        scanf("%d %d %d", &n, &m, &t);
        draw();
    }
}

int draw()
{
    if (t == 0 || t > n+m) {
        // e.g., 10 5 0
        for(int i=1; i<=n; i++) {
            printf("-");
        }
    }
    else if (t<=m) {
        // e.g., 10 5 1
        // e.g., 10 5 5
        for(int i=1; i<=n; i++) {
            if (i <= t) {
                printf("^");
            }
            else {
                printf("-");
            }
        }
    }
    else if (t<=n){
        // e.g., 10 5 6
        // e.g., 10 5 10
```

```
        // fix this section
        for(int i=1; i<=n; i++) {

            if(i >t-m && i <= t) {
                printf("^");
            }
            else {
                //5 3 4 n m t
                //t=4 m=3 n=5
                //i=1 "-"
                //i=2 "^"
                //i=3 "^"
                //i=4 "^"
                //i=5 "-"
                printf("-");
            }
        }
    }
    else if(t<=n+m){
        // e.g., 10 5 11
        // e.g., 10 5 12
        // fix this section
        for(int i=1; i<=n; i++) {
            if (i > t-m) {
                printf("^");
            }
            else {
                printf("-");
            }
        }
    }
    printf("\n");
}
```

More on `printf()` and `scanf()`



```
#include<stdio.h>
#define ENGINE 1499.99
int main(void)
{
    printf("~%f~\n", ENGINE);
    printf("~%e~\n", ENGINE);
    printf("~%4.2f~\n", ENGINE);
    printf("~%3.1f~\n", ENGINE);
    printf("~%10.3f~\n", ENGINE);
    printf("~%-10.3f~\n", ENGINE);
    printf("~%12.3e~\n", ENGINE);
    printf("~%+4.2f~\n", ENGINE);
    printf("~%010.2f~\n", ENGINE);
    return 0;
}
```

```
~1499.990000~
~1.499990e+03~
~1499.99~
~1500.0~
~  1499.990~
~1499.990  ~
~   1.500e+03~
~+1499.99~
~0001499.99~
```

Using the `%*.s` format in `printf()`

```
#include <stdio.h>
#include <string.h>
#define BORDER "##############################################"
int main(void)
{
    char word[26];
    scanf("%25s", word);
    printf("%.*s\n", strlen(word)+2, BORDER);
    printf("#%s#\n", word);
    printf("%.*s\n", strlen(word)+2, BORDER);

    return 0 ;
}
/*
Input: University
############
#University#
############
*/
```

'A' 與 65

```
#include <stdio.h>
int main(void)
{
    char ch;
    for (ch = 'a'; ch <= 'z'; ch++) {
        printf("The ASCII value for %c is %d.\n", ch, ch);
    }
    return 0;
}
```

用 EOF 判斷輸入是否結束

```
#include <stdio.h>

int main(void)
{
    int x;

    while (scanf("%d", &x) != EOF) { /*Ctrl Z + Enter */
        printf("x=%d\n", x);
    }
    return 0;
}
```

檔案的結尾一定會有 EOF 字元，因此可以用 EOF 判斷是否已經將檔案讀完

如果是用手動方式輸入，則要在行首打 Ctrl+Z 再按 Enter 就可以打出 EOF 字元

```
/*
輸入: ABCDE
輸出:
     A
    AB
   ABC
  ABCD
 ABCDE
*/
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

int main(void)
{
   int word_len;
   char word[40];
   scanf("%s", &word);
   word_len = strlen(word);

   for (int i = 1; i <= word_len; i++)
   {
      for( int k = 0; k<word_len-i;k++)
          {
            printf(" ");
          }
      for (int j = 0; j < i; j++)
      {
        printf("%c", word[j]);
      }
      printf("\n");
   }

   return 0;
}
```

getchar() and putchar()

寫程式時總會有需要一次只讀取或是輸出一個字元的時候，那這個時候就不能用scanf() 或是 printf()，而要用 getchar() 和 putchar()

`ctype.h  ex.tolower()  toupper()`

舉例：輸入一字串，然後1. 把空白字元拿掉 2. 把大寫換成小寫，小寫化成大寫 3. 數字了話，把原數字加2後輸出
ps.所謂的空白字元包含1. 空白鍵 2. tab鍵 3. 換行字元（enter鍵）

```
#include<stdio.h>
#include <ctype.h>

int main(void)
{
    char c;
    int n;
    while ( (c = getchar() ) != EOF )
    {
        if ( isspace(c)) continue;
        if ( isupper(c)) printf("%c\n", tolower(c) );
        if ( islower(c)) printf("%c\n", toupper(c) );
        if ( isdigit(c))
        {
            ungetc(c, stdin);
            scanf("%d", &n);
            printf("%d\n", n+2);
        }
    }
    return 0;
}
```

程式碼測試階段
可以在 `main` 一開始的地方加入底下的程式碼
這樣每次執行時就不必手動重複輸入測試資料

```
    freopen("testcase.txt", "r", stdin);
    freopen("output.txt", "w", stdout);
```

Transpose Matrix

```c
#include <stdio.h>

int main()
{
   int m, n, c, d, matrix[10][10], transpose[10][10];
   matrix[10][10] = {0};
   printf("Enter the number of rows and columns of matrix\n");
   scanf("%d%d", &m, &n);

   for (c = 0; c < m; c++)
      for( d = 0 ; d < n ; d++ )
         transpose[d][c] = matrix[c][d];

   printf("Transpose of entered matrix :-\n");

   for (c = 0; c < n; c++) {
      for (d = 0; d < m; d++)
         printf("%d\t",transpose[c][d]);
      printf("\n");
   }

   return 0;
}
```