Problem Solving (C37)

هذا البرنامج التدريبي مُصاغ بعناية لتمكين المتدربين من تطوير قدراتهم الفكرية على غرار المبرمجين المحترفين، والتعاون بكفاءة ضمن فريق محترف في شركة "جمال تك" أو أي مؤسسة متعددة الجنسيات أخرى. نظرًا لأهمية اللغة الإنجليزية في بيئة العمل العالمية، يتم تقديم المحتوى التدريبي بالإنجليزية. لا يشترط إتقان اللغة بشكل كامل، لكن من الضروري امتلاك القدرة الكافية لفهم المتطلبات وتنفيذها بشكل فعّال. يُمكن للمتدربين استخدام مترجم جوجل أو الاستعانة بـ "شات جي بي تي" للتغلب على أية عقبات لغوية، المهم هو الفهم الدقيق للمطلوب وتحقيقه بنجاح.

لتعظيم الاستفادة من التدريب، يُنصح بمحاولة حل التمارين بشكل مستقل لمدة ساعة واحدة على الأقل قبل الرجوع إلى الحل المرفق في نهاية الملف.

قد يتضمن الحل كودًا برمجيًا غير مفسر بعد، والغرض من ذلك هو تشجيعك على محاولة فهم الأكواد البرمجية الجديدة التي لم تتعرض لها من قبل. هذه المهارة ضرورية في سوق العمل، حيث تتطور لغات البرمجة باستمرار ويظهر كل يوم لغات جديدة. ستواجه دائمًا أكوادًا لم تدرسها من قبل، ومن المهم أن تكون قادرًا على فهمها بنفسك دون الحاجة إلى دراسة مسبقة. يمكنك الاستعانة بمحرك البحث جوجل، أو استخدام ChatGPT، أو حتى اللجوء لأصدقائك للمساعدة. الهدف الأساسي هو أن تصل إلى فهم معنى كل كود بأي طريقة ممكنة لتتمكن من إيجاد موقعك في سوق العمل.

إن وجود كود برمجي غير مفسر يشكل تحديًا يتوجب عليك إيجاد حل له. هذا النوع من التدريبات يعد جزءًا أساسيًا من تدريبات 'Problem Solving'، التي تهدف إلى تمكينك من أداء عملك بفاعلية بغض النظر عن التحديات والعقبات. هذه القدرة على حل المشكلات هي ما يتمتع به العاملون في 'جمال تك'، ومن الضروري أن تطور في نفسك هذه المهارة لتصبح عضوًا فعالًا في فريق عمل 'جمال تك'.

Gammal Tech's Happiness Booster

Background

Gammal Tech, a pioneer in the software development industry, is renowned for its cutting-edge office facilities, a highly innovative team, and an extraordinary work environment. Always at the forefront of innovation, Gammal Tech is exploring a unique system to enhance the happiness levels of its employees.

Problem Statement

Gammal Tech has developed a "Happiness Booster" system. This system uses a special algorithm to determine the optimal schedule for work and leisure activities for employees, thereby maximizing their happiness. Each employee has a "Happiness Level" represented by a unique positive integer. The system applies a "Happiness Modifier" using bitwise 'Left Shift' operations to these levels at different times of the day to maximize overall happiness.

The challenge is to design a part of this system. Given an initial happiness level of an employee and a sequence of 'Left Shift' operations throughout the day, you need to calculate the final happiness level at the end of the day.

Input Format

- The first line contains two integers `N` and `M`, where `N` is the number of employees and `M` is the number of 'Left Shift' operations applied to each employee in a day.
- The next `N` lines contain the initial happiness level of each employee, a positive integer `H`.
- The next `M` lines contain the number of positions `P` each 'Left Shift' operation will shift the happiness level.

Output Format

For each employee, output their final happiness level at the end of the day.

Constraints

- $1 \leq N \leq 1{,}000$
- $1 \leq M \leq 10$
- $1 \leq H \leq 1{,}000{,}000$
- $1 \leq P \leq 5$

Sample Input:

```
3 2
5
7
10
1
2
```

Sample Output:

```
20
28
40
```

Explanation

- For the first employee: Initial happiness level 5, after the first shift 10 and after the second shift 20.
- For the second employee: Initial happiness level 7, after the first shift 14 and after the second shift 28.
- For the third employee: Initial happiness level 10, after the first shift 20 and after the second shift 40.

لتحقيق أقصى فائدة من التدريب، يُوصى ببذل محاولة مستقلة لحل التمارين لمدة لا تقل عن ساعة واحدة. تجنب الاطلاع على بقية الملف حتى تكمل عملية التفكير في الحل. بعد ذلك، جرب حلك بنفسك على المدخلات الموضحة. إذا واجهت مدخلات لم تتوقعها، فهذا يعد فرصة لتطوير مهارة جديدة ضمن مسيرتك التعليمية. المهندس المحترف يجب أن يضمن أن برنامجه يعمل مع جميع أنواع المدخلات، وهذه مهارة يتم تطويرها عبر التجربة والخطأ. لذا، من الضروري ألا تطلع على المدخلات المتوقعة قبل أن تجرب الحل بنفسك. هذه هي الطريقة الأمثل لتنمية هذه المهارة.

بعد اختبار المدخلات المقترحة، إذا كانت النتائج تختلف عما هو مدون في الملف، فيُنصح بمحاولة حل التمرين مرة أخرى لمدة ساعة على الأقل قبل الرجوع إلى الحل الموجود في نهاية الملف.

Test Case 1: Moderate Number of Employees and Shifts

Input

```
5 3
10
20
30
40
50
2
1
3
```

Expected Output

```
80
160
240
320
400
```

Test the program's handling of a moderate number of employees and a sequence of shift operations.

Explanation:

- For the first employee: Initial happiness level 10, after the first shift 40, second shift 80, and third shift $80 \ll 3 = 640$.
- Similar calculations apply for the other employees, with their respective initial happiness levels.

Test Case 2: Maximum Happiness Level and Maximum Shift

Input

```
1 1
1000000
5
```

Expected Output

```
32000000
```

Test the program with the maximum happiness level and the maximum shift.

Explanation: $1000000 \ll 5 = 32000000$

Test Case 3: Minimum Happiness Level and Minimum Shift

Input

```
1 1
1
1
```

Expected Output

```
2
```

Test the program with the minimum happiness level and minimum shift.

Explanation: $1 \ll 1 = 2$

Test Case 4: Alternating Shift Values

Input

```
2 3
4
8
1
2
1
```

Expected Output

```
32
64
```

Test how the program handles alternating shift values.

Explanation: For the first employee: $4 \ll 1=8$, $8 \ll 2=32$, $32 \ll 1=64$

Test Case 5: No Shift Operation

Input

```
1 1
12345
0
```

Expected Output

```
12345
```

Ensure that a zero shift leaves the happiness level unchanged.

Explanation: $12345 \ll 0 = 12345$

لتحقيق أقصى استفادة من التدريب، من المستحسن أن تخصص وقتًا إضافيًا - لا يقل عن ساعة - لمحاولة حل التمرين مرة أخرى بمفردك قبل الرجوع إلى الحل المقترح. هذه العملية المتكررة من التجربة والخطأ تعتبر استراتيجية فعالة في تعزيز مهاراتك البرمجية وتعميق فهمك للمفاهيم. تذكر أن التحدي والمثابرة هما المفتاحان للتطور في مجال البرمجة.

C Programming Solution:

```c
#include <stdio.h>

int main() {
    int N, M;
    scanf("%d %d", &N, &M);

    int happinessLevels[N], shiftPositions[M];

    // Reading initial happiness levels
    for(int i = 0; i < N; i++) {
        scanf("%d", &happinessLevels[i]);
    }

    // Reading shift positions
    for(int i = 0; i < M; i++) {
        scanf("%d", &shiftPositions[i]);
    }

    // Calculating final happiness levels
    for(int i = 0; i < N; i++) {
        int finalLevel = happinessLevels[i];
        for(int j = 0; j < M; j++) {
            finalLevel <<= shiftPositions[j]; // Applying left shift
        }
        printf("%d\n", finalLevel);
    }

    return 0;
}
```