Problem Solving (C35)

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 Galactic Expansion

Background: Gammal Tech, a leader in the software development industry, known for its innovative solutions and exceptional work environment, has embarked on an ambitious project. With its reputation for pushing boundaries, Gammal Tech now aims to establish a new branch on a newly discovered planet in the universe. This venture requires a robust system design to ensure the success of this interstellar expansion.

Task: As a lead engineer at Gammal Tech, you are tasked with designing a software system to select the most suitable planet for the new branch. The system must analyze data from various planets and choose the one that meets specific criteria.

Data: Each planet is represented by a unique 32-bit integer identifier. The suitability of a planet is determined by certain characteristics encoded in specific bit positions within this identifier:

- Bits 0-7: Climate Score (0-255)

- Bits 8-15: Technology Adaptability Index (0-255)
- Bits 16-23: Resource Availability Score (0-255)
- Bits 24-31: Distance Factor (0-255)

Criteria: A planet is considered suitable if it satisfies the following conditions:

- The bitwise OR of the Climate Score and Technology Adaptability Index must be greater than 200.
- The bitwise OR of the Resource Availability Score and Distance Factor must be less than 100.

Input: The first line contains an integer `N`, the number of planets to analyze. The following `N` lines each contain a 32-bit integer representing a planet.

Output: Print the identifier of the most suitable planet. If no suitable planet is found, print "No suitable planet found".

Sample Input:

```
3
4294967295
2155872255
3232235776
```

Sample Output:

```
2155872255
```

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: No Suitable Planet

Input

```
3
10000000
20000000
30000000
```

Expected Output

```
No suitable planet found
```

Explanation: None of the planets meet the criteria for suitability.

Test Case 2: All Planets Suitable

Input

```
3
4294967295
2155872255
3232235776
```

Expected Output

```
4294967295
```

Explanation: All planets meet the criteria, and the planet with the identifier `4294967295` has the highest total score.

Test Case 3: Single Planet with Edge Values

Input

```
1
255255255255
```

Expected Output

```
255255255255
```

Explanation: The single planet meets the criteria, and its identifier is the output.

Test Case 4: Large Number of Planets with No Suitable Options

Input

```
5
11111111
22222222
33333333
44444444
55555555
```

Expected Output

```
No suitable planet found
```

Explanation: Although there are multiple planets, none of them meet the suitability criteria.

Test Case 5: Planets with Same Scores

Input

```
4
4294967295
4294967295
3232235776
2155872255
```

Expected Output

```
4294967295
```

Explanation: Two planets have the same highest score and meet the criteria, so the first one encountered (with identifier `4294967295`) is chosen as the output.

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;
    scanf("%d", &n);

    unsigned int planet, bestPlanet = 0;
    int maxScore = -1;

    for (int i = 0; i < n; i++) {
        scanf("%u", &planet);

        int climateScore = planet & 0xFF;
        int techIndex = (planet >> 8) & 0xFF;
        int resourceScore = (planet >> 16) & 0xFF;
        int distanceFactor = (planet >> 24) & 0xFF;

        if ((climateScore | techIndex) > 200 && (resourceScore |
distanceFactor) < 100) {
            int totalScore = climateScore + techIndex + resourceScore +
distanceFactor;
            if (totalScore > maxScore) {
                maxScore = totalScore;
                bestPlanet = planet;
            }
        }
    }

    if (bestPlanet) {
        printf("%u\n", bestPlanet);
    } else {
        printf("No suitable planet found\n");
    }

    return 0;
}
```