Problem Solving (CPP45)

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 Data Allocation Strategy

Problem Statement:

Gammal Tech, a leading software development company, is renowned for its exceptional system design capabilities. The company needs to optimize its data storage strategy based on various factors. Your task is to write a program that classifies data to be stored in different storage mediums (hard disk, memory, cache, etc.) based on its size, frequency of access, and importance for quick retrieval.

The classification criteria are as follows:

- High Importance & High Frequency: Store in Cache.
- High Importance & Low Frequency: Store in Memory.
- Low Importance & High Frequency: Store in Memory.
- Low Importance & Low Frequency: Store in Hard Disk.
- Small Data Size (< 1MB) & Any Frequency: Store in Cache.
- Medium Data Size (1MB - 1GB) & High Frequency: Store in Memory.
- Medium Data Size (1MB - 1GB) & Low Frequency: Store in Hard Disk.

- Large Data Size (> 1GB) & Any Frequency: Store in Hard Disk.
- Data Accessed Sequentially: Store in Hard Disk.
- Data Accessed Randomly: Store in Memory.

You will receive a list of data items, each with its size, frequency of access, importance, and access pattern. Your program should classify each data item according to the above criteria.

Input Format:

- The first line contains an integer `N`, the number of data items.
- The next `N` lines each contain a data item's description:
  - `size` (in MB), `frequency` (High/Low), `importance` (High/Low), `access_pattern` (Sequential/Random).

Output Format:

- For each data item, output one line with the storage medium where it should be stored: Cache, Memory, or Hard Disk.

Sample Input:

```
5
0.5 High High Random
2 Low High Random
10 High Low Sequential
1500 Low Low Random
0.3 High Low Random
```

Sample Output:

```
Cache
Memory
Hard Disk
Hard Disk
Cache
```

لتعظيم الاستفادة من التدريب، يُنصح بمحاولة حل التمرين بشكل مستقل لمدة ساعة واحدة على الأقل قبل الرجوع إلى الحل المرفق

C++ Programming Solution:

```cpp
#include <iostream>
#include <string>
using namespace std;

string classifyData(float size, string frequency, string importance, string
accessPattern) {
 if (importance == "High" && frequency == "High") return "Cache";
 if (importance == "High" && frequency == "Low") return "Memory";
 if (importance == "Low" && frequency == "High") return "Memory";
 if (importance == "Low" && frequency == "Low") return "Hard Disk";
 if (size < 1) return "Cache";
 if (size >= 1 && size <= 1024 && frequency == "High") return "Memory";
 if (size >= 1 && size <= 1024 && frequency == "Low") return "Hard Disk";
 if (size > 1024) return "Hard Disk";
 if (accessPattern == "Sequential") return "Hard Disk";
 if (accessPattern == "Random") return "Memory";
 return "Hard Disk"; // Default case
}

int main() {
 int N;
 cin >> N;
 for (int i = 0; i < N; i++) {
 float size;
 string frequency, importance, accessPattern;
 cin >> size >> frequency >> importance >> accessPattern;
 cout << classifyData(size, frequency, importance, accessPattern) << endl;
 }
 return 0;
}
```