????

????

C#???????

```
int Sum(int a, int b)
{
    return a + b;
}
```

□□□

□□□□□□　　a□ b□□□□□□□□□□□□□□□□□□

```
int Max(int a, int b)
{
    if (a > b)
        return a;
    else
        return b;
}
```

□□

□□□□□□　　a□ b□□□□□□□□□□□□□□□□□□

```
bool CheckHasItem(string[] itemList, string item)
{
    for (int i = 0; i < itemList.Length; i++)
    {
        if (itemList[i] == item)
            return true;
    }
```

```
    return false;
}
```

答え

「itemList」配列の中に、「item」という文字列が入っているかどうかを確認し、あったら「true」を返し、なければ、「false」を返す。つまり、アイテム存在しているかどうかを確認。

Unity???

1. 配列で作った「最も長いフルーツ_____」のスクリプトを更新し、「`FindLongString`」というメソッドにしよう。配列を引数として渡し、最も長い文字列を返すようにてください。

使い方の例:

```
void Start()
{
    string [] fruits = {"lemon", "mikan", "orange", "grapefruit", "kiwi"};

    // このメソッドを作ってください
    string maxFruit = FindLongString(fruits);

    // grapefruitを表示されるはず
    Debug.Log($"最も長いフルーツ名は: {maxFruit}");
}

// ここからメソッドを作る...
```

答え

```
// 最も長い文字列を探索
string FindLongString (string [] fruits)
{
  // 最も長い文字列
    string maxFruit = "";
```

```
    // フルーツ1個ずつ確認
    foreach(string f in fruits)
    {
        // もし、今回のフルーツ名の方が長ければ
        if (f.Length > maxFruit.Length)
        {
            // 最も長いフルーツを更新
            maxFruit = f;
        }
    }

    // 結果を返す
    return maxFruit;
}
```

2.「じゃんけん」ゲーム結果の判定メソッド「`CheckJanken`」を作成してみてください。

前提条件:

- グーは「0」、チョキは「1」、パーは「2」として表現する
- グー、チョキ、パー（0, 1, 2）以外の数値入れたら、エラーを返す
- プレーヤーは2人（AとB）
- 判定の結果は: Aの勝ち: 1、Bの勝ち: 2、ひきわけ: 0、エラー: -1
- メソッド名: `CheckJanken`
  - 引数: 各プレーヤーの選択肢
  - 戻り値: 判定の結果

使い方の例1（Aの勝ち）

```
void Start()
{
  int playerA = 1; // チョキ
  int playerB = 2; // パー
  int result = CheckJanken(playerA, playerB);// 1を返す（Aの勝ち）
  Debug.Log (result)
}
```

使い方の例2（Bの勝ち）

```
void Start()
{
  int playerA = 0; // グー
  int playerB = 2; // パー
  int result = CheckJanken(playerA, playerB);　// 2を返す（Bの勝ち）
  Debug.Log(result);
}
```

使い方の例3（あいこ）

```
void Start()
{
  int playerA = 2; // パー
  int playerB = 2; // パー
  int result = CheckJanken(playerA, playerB);　// 0を返す（あいこ）
  Debug.Log(result);
}
```

使い方の例4（エラー）

```
void Start()
{
  int playerA = 4; // ???
  int playerB = 2; // パー
  int result = CheckJanken(playerA, playerB);　// -1を返す（エラー）
  Debug.Log(result);
}
```

答え

```
// じゃんけんの結果の判定を求める
// playerA: プレーヤーAの選択肢（0:グ, 1:チョキ, 2:パー）
// playerB: プレーヤーBの選択肢（0:グ, 1:チョキ, 2:パー）
// 戻り値:
//  -1: エラー（無効な選択肢）
//   0: あいこ
//   1: プレーヤーAの勝ち
//   2: プレーヤーAの勝ち
int CheckJanken (int playerA, int playerB)
{
```

```
    // □□□□□□□□□□□□□□A□
    if (playerA < 0 || playerA > 2)
    {
        return -1;
    }
  □// □□□□□□□□□□□□□□B□
    if (playerB < 0 || playerB > 2)
    {
        return -1;
    }

    // □□□
    if (playerA == playerB)
    {
        return 0;
    }

    // □□□□□A□□□
    if ((playerA == 0 && playerB == 1) || (playerA == 1 && playerB == 2) || (playerA == 2
&& playerB == 0))
    {
        return 1;
    }

    // □□□□□B□□□□□□□□□□□□
    return 2;
}
```

Revision #3
Created 2026-04-07 02:42:42 UTC by Menendez Francisco
Updated 2026-04-28 01:43:49 UTC by Menendez Francisco