CodeName Code

Action Script package {

```
    import flash.display.Sprite;
    import flash.display.Bitmap;
    import flash.display.BitmapData;
    import flash.display.Loader;
    import flash.net.URLRequest;
    import flash.events.Event;
    import flash.geom.Point;
    import flash.geom.Rectangle;
    public class Main extends Sprite {
        private var _bitmap:BitmapData= new BitmapData(stage.stageWidth,stage.stageHeight,false,
0xffffffff);
        private var _loader:Loader = new Loader( );
        public function Main( ) {
            _loader.contentLoaderInfo.addEventListener(Event.COMPLETE, onLoad);
            _loader.load(new URLRequest("m.jpg"));
            var image:Bitmap = new Bitmap(_bitmap);
            addChild(image);
        }
        public function onLoad(event:Event):void {
            var loaderBmp:Bitmap = Bitmap(_loader.content);
            var w:Number = loaderBmp.width / 5;
            for(var i:int = 0; i < 10; i++) {
                _bitmap.copyPixels(loaderBmp.bitmapData,
                               new Rectangle(i * w, 0,
                                             w, loaderBmp.height),
                               new Point(i * (w + 2), i));
            }
        }
    }
}
C           #include <stdio.h>
struct Distance {
  int feet;
  float inch;
} d1, d2, result;
int main() {
  // take first distance input
  printf("Enter 1st distance\n");
  printf("Enter feet: ");
  scanf("%d", &d1.feet);
  printf("Enter inch: ");
  scanf("%f", &d1.inch);
  // take second distance input
  printf("\nEnter 2nd distance\n");
  printf("Enter feet: ");
  scanf("%d", &d2.feet);
  printf("Enter inch: ");
  scanf("%f", &d2.inch);
  // adding distances
  result.feet = d1.feet + d2.feet;
  result.inch = d1.inch + d2.inch;
  // convert inches to feet if greater than 12
  while (result.inch >= 12.0) {
    result.inch = result.inch - 12.0;
    ++result.feet;
  }
```

```
   printf("\nSum of distances = %d\'-%.1f\"", result.feet, result.inch);
   return 0;
}
C#         using System;
class Multipication
{
    static void Main()
    {
        int no;
        Console.Write("Enter a no : ");
        no = Convert.ToInt32(Console.ReadLine());
        while (no <= 0)
        {
            Console.WriteLine("You entered an invalid no");
            Console.Write("Enter a no great than 0: ");
            no = Convert.ToInt32(Console.ReadLine());
        }
        Console.WriteLine("Multiplication Table :");
        for (int i = 1; i <= no; i++)
        {
            Console.WriteLine("\n");
            for (int j = 1; j <= no; j++)
            {
                Console.Write("{0,6}", i * j);
            }
        }
        Console.Read();
    }
}
```

```
using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

namespace ConsoleApplication

{

  class Program

  {

    static void Main(string[] args)

    {

      int b1, b2;

      int i = 0, rem = 0;

      int[] sum = new int[20];

      Console.WriteLine("Enter the first binary number: ");

      b1 = int.Parse(Console.ReadLine());

      Console.WriteLine("Enter the second binary number: ");

      b2 = int.Parse(Console.ReadLine());

      while (b1 != 0 || b2 != 0)

      {

        sum[i++] = (b1 % 10 + b2 % 10 + rem) % 2;

        rem = (b1 % 10 + b2 % 10 + rem) / 2;

        b1 = b1 / 10;

        b2 = b2 / 10;

      }

      if (rem != 0)

        sum[i++] = rem;

      --i;

      Console.WriteLine("Sum of two binary numbers: ");

      while (i >= 0)
```

```
                Console.Write("{0}", sum[i--]);
            Console.ReadLine();
        }
    }
}
C++       #include <bits/stdc++.h>
using namespace std;
// Function to calculate x
// raised to the power y
int power(int x, unsigned int y)
{
    if (y == 0)
        return 1;
    if (y % 2 == 0)
        return (power(x, y / 2) * power(x, y / 2));
    return (x * power(x, y / 2) * power(x, y / 2));
}
// Function to calculate
// order of the number
int order(int x)
{
    int n = 0;
    while (x) {
        n++;
        x = x / 10;
    }
    return n;
}
// Function to check whether the
// given number is Armstrong number
// or not
bool isArmstrong(int x)
{
    // Calling order function
    int n = order(x);
    int temp = x, sum = 0;
    while (temp) {
        int r = temp % 10;
        sum += power(r, n);
        temp = temp / 10;
    }
    // If satisfies Armstrong
    // condition
    return (sum == x);
}
// Driver code
int main()
{
    int x = 153;
    cout << boolalpha << isArmstrong(x) << endl;
    x = 1253;
    cout << boolalpha << isArmstrong(x) << endl;
    return 0;
}
```