Soal 1

1. How about your opinion..?

```
if (application != null)
{
   if (application.protected != null)
   {
      return application.protected.shieldLastRun;
   }
}
```

Key:
cleaner and easier to read code.

Jawab :
1. Pernyataan if (application != null) memeriksa apakah objek aplikasi bukan null. Jika null, kode selanjutnya tidak akan dieksekusi.

2. Jika objek aplikasi bukan null, kontrol memasukkan pernyataan if bagian dalam: if (application.Protected != null). Ini memeriksa apakah properti Dilindungi dari objek aplikasi tidak null. Sekali lagi, jika null, kode selanjutnya tidak akan dieksekusi.

3. Dengan asumsi 2 step diatas terpenuhi (yaitu, aplikasi bukan null dan aplikasi. Dilindungi bukan null), kode mengembalikan nilai properti shieldlastrun dari objek yang Dilindungi.

Soal 2

2. How about your opinion.

```
public ApplicationInfo GetInfo()
{
   var application = new ApplicationInfo
   {
      Path = "C:/apps/",
      Name = "Shield.exe"
   };
   return application;
}
```

Key:
return more than one value from a class method.

Jawab :

1. Ini mendeklarasikan metode bernama GetInfo() dengan tipe kembalian ApplicationInfo. Kata kunci public menunjukkan bahwa metode ini dapat diakses dari luar

2. Di dalam metode ini, membuat instance baru dari kelas ApplicationInfo menggunakan kata kunci baru.

menginisialisasi properti objek ApplicationInfo. Properti Path diatur ke "C:/apps/", dan properti Nama diatur ke "Shield.exe".

Terakhir, ia mengembalikan objek ApplicationInfo yang baru dibuat.

Singkatnya, saat memanggil metode GetInfo(), metode ini akan membuat dan mengembalikan objek ApplicationInfo dengan jalur "C:/apps/" dan nama "Shield.exe". Metode ini berguna untuk memperoleh informasi tentang suatu aplikasi, seperti direktori instalasi dan nama file yang dapat dieksekusi.

Soal 3

3. How about your opinion.

```
class Laptop
{
        public string Os{ get; set; } // can be modified
        public Laptop(string os)
        {
                Os= os;
        }
}
var laptop = new Laptop("macOs");
Console.WriteLine(Laptop.Os); // Laptop os: macOs
```

Jawab :

Pembuatan Objek Laptop:

Sebuah instance dari kelas Laptop dibuat menggunakan kata kunci new, dengan sistem operasi "macOs" diteruskan sebagai argumen ke konstruktor.

Contoh ini mewakili objek laptop tertentu dengan sistem operasi yang disetel ke "macOs".

Output:

Properti Os dari instance laptop diakses menggunakan notasi titik (laptop.Os), dan nilainya dicetak ke konsol.

Outputnya adalah: Laptop os: macOs, menunjukkan sistem operasi laptop yang dibuat, yaitu "macOs" dalam hal ini.

Soal 4

4. How about your opinion?

```
using System;
using System.Collections.Generic;

namespace MemoryLeakExample
{
    class Program
    {
        static void Main(string[] args)
        {
            var myList = new List();
            while (true)
            {
                // populate list with 1000 integers
                for (int i = 0; i < 1000; i++)
                {
                    myList.Add(new Product(Guid.NewGuid().ToString(), i));
                }
                // do something with the list object
                Console.WriteLine(myList.Count);
            }
        }
    }

    class Product
    {
        public Product(string sku, decimal price)
        {
            SKU = sku;
            Price = price;
        }

        public string SKU { get; set; }
        public decimal Price { get; set; }
    }
}
```

Key:
Keeping references to objects unnecessarily

Jawab :

1. myList akan mendapat kan nilai guid Metode statis ini membuat GUID baru. Setiap kali dipanggil, ini menghasilkan pengidentifikasi unik.

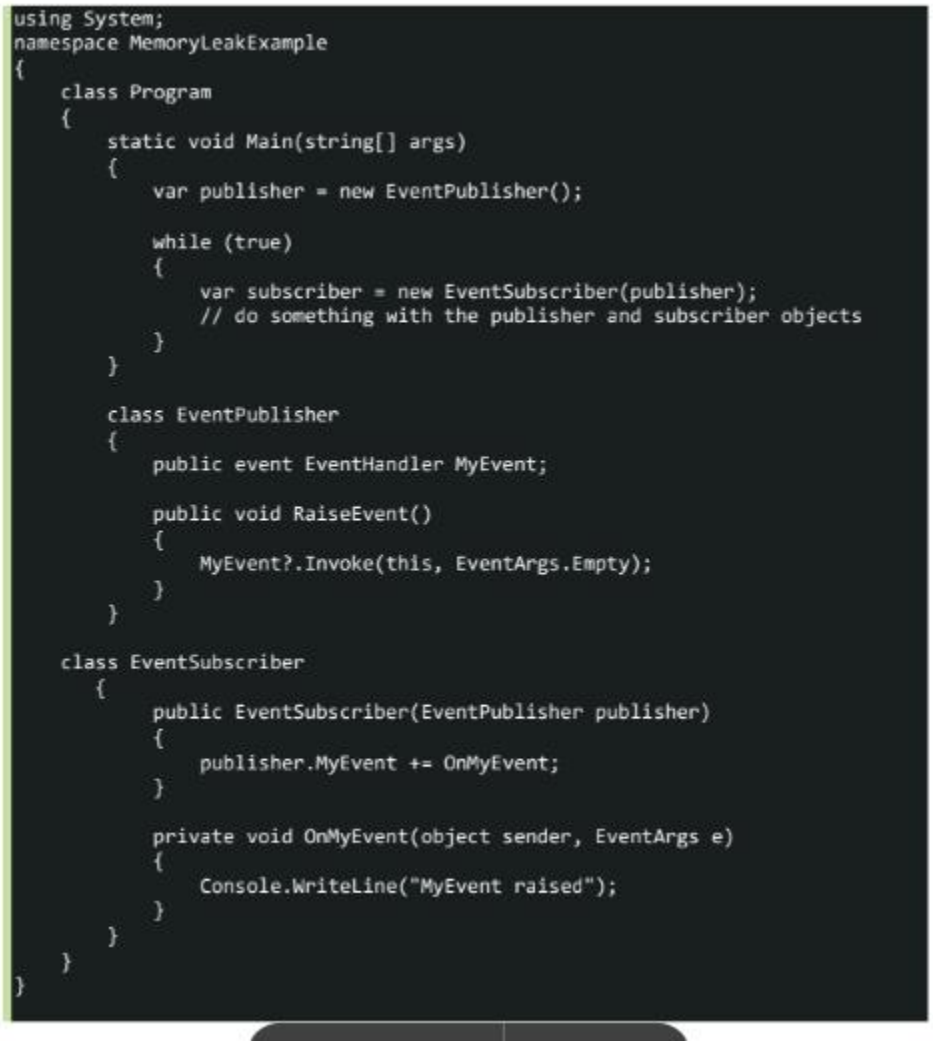
Console write line akan menghitung nilai mylist.count ada loop lain yang mengisi daftar myList dengan 1000 objek Produk.

2.Ini memulai perulangan tak terbatas.

3.Class Product menunjukan object para product dengan property string SKU, Decimal Price

Soal 5

5. How about your opinion?



```
using System;
namespace MemoryLeakExample
{
    class Program
    {
        static void Main(string[] args)
        {
            var publisher = new EventPublisher();

            while (true)
            {
                var subscriber = new EventSubscriber(publisher);
                // do something with the publisher and subscriber objects
            }
        }

        class EventPublisher
        {
            public event EventHandler MyEvent;

            public void RaiseEvent()
            {
                MyEvent?.Invoke(this, EventArgs.Empty);
            }
        }

    class EventSubscriber
       {
            public EventSubscriber(EventPublisher publisher)
            {
                publisher.MyEvent += OnMyEvent;
            }

            private void OnMyEvent(object sender, EventArgs e)
            {
                Console.WriteLine("MyEvent raised");
            }
        }
    }
}
```

Jawab :
Method Main

1. menciptakan sebuah instance dari EventPublisher bernama penerbit.

Di dalam loop while yang tak terbatas, berulang kali membuat instance EventSubscriber baru, meneruskan objek penerbit ke setiap pelanggan

2. fungsi EventHandler.

Ia memiliki metode bernama RaiseEvent() yang memunculkan MyEvent. Metode ini memanggil kejadian jika bukan null menggunakan operator kondisional null

Kelas ini mengambil instance EventPublisher sebagai parameter konstruktor.

Di dalam konstruktor, ia berlangganan acara MyEvent dari instance EventPublisher yang disediakan dengan menambahkan metode pengendali OnMyEvent ke acara tersebut

3.Metode ini (OnMyEvent) dipanggil ketika MyEvent dimunculkan.

Itu hanya menulis "MyEvent raise" ke konsol.

4. Metode Main berisi perulangan while (true), yang berjalan tanpa batas.

Di dalam loop ini, instance EventSubscriber baru dibuat pada setiap iterasi.

Setiap instance EventSubscriber berlangganan acara MyEvent dari EventPublisher yang sama.

6. How about your opinion?

```
using System;
using System.Collections.Generic;
namespace MemoryLeakExample
{
    class Program
    {
        static void Main(string[] args)
        {
            var rootNode = new TreeNode();
            while (true)
            {
                // create a new subtree of 10000 nodes
                var newNode = new TreeNode();
                for (int i = 0; i < 10000; i++)
                {
                    var childNode = new TreeNode();
                    newNode.AddChild(childNode);
                }
                rootNode.AddChild(newNode);
            }
        }
    }

    class TreeNode
    {
        private readonly List<TreeNode> _children = new List<TreeNode>();
      public void AddChild(TreeNode child)
      {
            _children.Add(child);
        }
    }
}
```

Key:
Large object graphs

Jawab :

1. Main Method berfungsi sebagai titik masuk program.Ini menciptakan simpul akar (rootNode) untuk struktur pohon.

2.Di dalam perulangan while (true) yang tak terbatas, kode tersebut berulang kali membuat subpohon baru dan menambahkannya ke simpul akar.

Setiap subpohon terdiri dari 10.000 node.

3.Membuat Node:

Untuk setiap iterasi loop, sebuah node baru (newNode) dibuat.

Di dalam loop, 10.000 node anak dibuat (childNode) dan ditambahkan ke node baru menggunakan metode AddChild.

Node baru (newNode) kemudian ditambahkan ke node root (rootNode) menggunakan metode AddChild

4.Class Node Pohon:

Class ini mewakili sebuah node dalam struktur pohon.

Ini berisi daftar pribadi _children untuk menyimpan referensi ke node turunannya.

Metode AddChild memungkinkan penambahan node anak ke node saat ini dengan menambahkannya ke daftar _children.

Soal 7

7. How about your opinion?

```
using System;
using System.Collections.Generic;
class Cache
{
  private static Dictionary<int, object> _cache = new Dictionary<int,
object>();

  public static void Add(int key, object value)
  {
    _cache.Add(key, value);
  }

  public static object Get(int key)
  {
    return _cache[key];
  }
}

class Program
{
  static void Main(string[] args)
  {
    for (int i = 0; i < 1000000; i++)
    {
      Cache.Add(i, new object());
    }

    Console.WriteLine("Cache populated");

    Console.ReadLine();
  }
}
```

Jawab :

mekanisme caching dengan code C #

Class cache:

bertindak sebagai cache.

memiliki bidang statis pribadi _cache, yang merupakan kamus dengan kunci bilangan bulat (int) dan nilai objek (objek). Kamus ini menyimpan data cache.

Ini memiliki dua metode statis:

Add(int key, object value): Menambahkan pasangan kunci-nilai ke cache.

Get(int key): Mengambil nilai yang terkait dengan kunci yang diberikan dari cache.

Pro