learnBIG Homework Assignment

Thank you for your interest in becoming a potential BIGgie!

As part of our hiring process, we sometimes ask candidates applying for software development positions to complete a short pre-interview "homework" task. This task is designed to give us insight into your software development skills and experience, and we appreciate you taking the time to complete it.

The task we are asking you to complete involves implementing a class called “NumberPool” which is described below. The basic outline of the class is provided in three languages: C++, C#, and Java. You should implement the solution in **just one** of these languages, or any other language of your choice.

Your solution should be complete and free from syntax errors, so the evaluator can build and run it.

The NumberPool Class

The **NumberPool** class represents the pool of numbers ranging from one to ten-million (1 to 10,000,000). This class has a default constructor and, in the case of C++, a destructor. It also contains two additional public methods: **Allocate** and **Release**.

The **Allocate** method picks an available value from the pool, removes it from the pool, and returns this value to the caller. If the pool of available numbers is empty, the Allocate method returns 0.

The **Release** method adds an available number value back to the pool. If the value is successfully added back to the pool, **Release** returns true. If the value is already in the pool, then false is returned.

When an instance of the NumberPool is instantiated, the object state is such that the range of numbers (1 to 10,000,000) is in the pool. The code can call the Allocate method and expect it to succeed immediately after the object is created.

Your task is to implement the NumberPool class. On the following pages are class declarations in C++, C#, and Java. These class declarations contain "stubbed out" implementations for the Allocate and Release methods as well as for the language-appropriate constructors and destructors. Your design criteria are the following:

> You may add any additional member methods or variables to the NumberPool class declaration. You may define additional functions and data structures if needed.
>
> Instances of the NumberPool class will run within a single thread. That is, you don't have to worry about threads or concurrency issues.
>
> If implementing the class in C++, you may use common functions from the standard C and C++ libraries. Do not use any external container libraries such as those from ATL, MFC, or Boost.
>
> If implementing the class in C# or Java, you may use classes and functions for the .NET and Java SE class libraries respectively, but you may *not* use any external libraries which are not distributed as part of the C# or Java standard distributions.

Be thoughtful and efficient with memory in the average case. A solution involving an array of 10 million Boolean values is not considered efficient with memory. It may be possible for an application to "Allocate" all ten million available numbers from the pool, but this may be an extreme case. In the average case, far fewer will be allocated, but the allocation/release call patterns may vary widely.

Can you keep the memory allocation under 1 MB in the average case?

Can you make your code scale to handle the extreme case while still keeping good run-time performance and reasonable memory usage?

Along with your solution, please also provide your thoughts on the following:

What other solutions did you consider and why did you select the presented solution?

What are the pros and cons or tradeoffs of your solution?

What tradeoffs did you choose not to make?

How would you modify your solution if memory was not an issue? Why?

How would you characterize the performance of your implementation in the average and extreme cases?

Number Set – C++

```
class NumberPool
{
public:
    NumberPool();
    ~NumberPool();

    int Allocate();
    bool Release(int x);
};


NumberPool::NumberPool()
{

    // Add your own code here

};

NumberPool::~NumberPool()
{

     // Add your own code here

}

int NumberPool::Allocate()
{

    // Add your own code here

}

bool NumberPool::Release(int x)
{

    // Add your own code here

}
```

NumberPool – C#

```
class NumberPool
{
    public NumberPool()
    {
        // Add your own code here
    }

    public int Allocate()
    {
        // Add your own code here
    }

    public bool Release(int x)
    {
      // Add your own code here
    }
}
```

NumberPool – Java

```java
class NumberPool
{
    public NumberPool()
    {
        // Add your own code here
    }

    public int Allocate()
    {
        // Add your own code here
    }

    public boolean Release(int x)
    {
        // Add your own code here
    }
}
```