The Art Institute of Vancouver℠

Course Number: **VGP336**
Course Title: **Gameplay Programming**

Assignment #1: Block Allocator

Date Assigned: Week 2

Due Date: Week 3

Deliverable:
Assignment will be collected in-class via the student shared Assignments folder.

Marks Breakdown:
Assignments are worth 70% of the final course mark. Late assignments will not be accepted.

Description:

A Block Allocator can improve performance over standard new/delete as it provides constant time allocation/deallocation. One can also prevent dangling pointers and memory stomps if the memory is access via a weak handle abstraction. You task is the finish the implementation of **BlockAllocator** class and **Handle** class and make sure they pass all test cases in the provided unit tests.

BlockAllocator.h

```
#ifndef INCLUDED_ENGINE_BLOCKALLOCATOR_H
#define INCLUDED_ENGINE_BLOCKALLOCATOR_H

#include "Handle.h"

template <typename T>
class BlockAllocator
{
public:
	BlockAllocator(u16 capacity);
	~BlockAllocator();

	Handle<T> New();
	void Delete(Handle<T> handle);

	bool IsValid(Handle<T> handle) const;

	T* Get(Handle<T> handle);

private:
	s32 mFreeSlot;
	u16 mCapacity;
	T* mData;
	u16* mGenerations;
};

#include "BlockAllocator.inl"

#endif // #ifndef INCLUDED_ENGINE_BLOCKALLOCATOR_H
```

Handle.h

```
#ifndef INCLUDED_ENGINE_HANDLE_H
#define INCLUDED_ENGINE_HANDLE_H

template <typename T> class BlockAllocator;

template <typename T>
class Handle
{
public:
	Handle();

	bool IsValid() const;
	void Invalidate();

	T* Get();

	u16 GetIndex() const			{ return mIndex; }
	u16 GetGeneration() const	{ return mGeneration; }

private:
	friend class BlockAllocator<T>;

	static BlockAllocator<T>* sAllocator;

	u16 mIndex;
	u16 mGeneration;
};

#include "Handle.inl"

#endif // #ifndef INCLUDED_ENGINE_HANDLE_H
```