实验三：C 内存管理程序设计

一、实验目标

1. 第四章-C 内存管理；

2. 了解动态内存管理的基本原理；

3. 掌握动态内存分配函数的使用方法。

二、内容与要求

1. 所在模块：**MemPool**

2. 编写一个简易的内存池管理模块，采用 C 内存管理函数实现内存池的管理。内存池按块进行管理，每块内存为 256 字节。MemPool 完成内存池的初始化、内存分配、内存回收。内部实现可自由发挥，但函数接口必须一致。

3. 编写函数：**int initPool(size_t size = 1000);**

功能：分配初始内存池，内存池初始大小默认为 1000 个块，即 256 x 1000 字节。

参数 size：内存池初始化时分配的内存块个数，默认为 1000 块。

返回值：成功完成 1000 块内存的申请则返回 0，否则返回-1.

提示：可使用 malloc、calloc 等函数完成内存初始化。

4. 编写函数：**char* allocBlock();**

功能：从内存池中找到一块空闲内存块进行分配。分配的内存被初始化为 0。**如果内存池内存已分配完，需对现有内存池进行扩充。**

返回值：返回分配的内存块首地址，如果出现错误则返回空指针。

提示：可使用 malloc、calloc 函数、memset 函数。注意如果使用 realloc 扩充内存，通过 **allocBlock** 已经分配出去的内存块地址可能会无效，导致无法回收内存块。

5. 编写函数：**int freeBlock(char* buf);**

功能：将 buf 指向的内存块释放，该内存块并不执行堆内存的释放操作，只是放回内存池供下次分配使用。释放时，需对 buf 指向的内存区域**进行清零操作**。

参数 buf：指向要放回内存池中的内存块。

返回值：执行成功返回 0；buf 指针参数为空返回-1；根据 buf 指针未找到对应的块，即 buf 指针并不是以前 **allocBlock** 函数分配的内存块地址，则返回-2.

提示：可使用 memset 函数，或者 Windows API SecureZeroMemory 函数清零。

6. 编写函数：**int freePool();**

功能：将整个内存池的内存全部释放，首先判断是否存在未回收的内存块，如果有则需返回-1，如果没有则对内存池所有内存块进行清零，然后执行 free 操作。.

返回值：执行成功返回 0，如果存在未回收的内存块，则需返回-1.

提示：可使用 free、memset 函数。

7. 编写函数：**int freePoolForce();**

功能：无论是否存在未回收的内存块，都将整个内存池的内存块进行清零，然后执行 free 操作。.

返回值：执行成功返回 0，出现错误，则返回-1.

提示：可使用 free、memset 函数。

8. 编写函数：**size_t getBlockCount();**

功能：返回内存池中所有的内存块个数。

9. 编写函数：**size_t getAvailableBlockCount();**

功能：返回内存池中空闲可分配的内存块个数。

10. 在 main 函数中对所编写函数进行测试，说明所使用的测试用例。

11. MSDN: https://msdn.microsoft.com/en-us/library/hh875057.aspx

12. 建议学时：2 学时。

三、参考测试用例

序号	
1	**initPool(8);allocBlock();char* buf = allocBlock();** **buf[0] + buf[1] + ... +buf[255] = 0**
2	**getBlockCount()==8;getAvailableBlockCount()==6;** **allocBlock();allocBlock();getAvailableBlockCount()==4;**
3	**freeBlock(buf);getBlockCount()==8;getAvailableBlockCount()==5;**
4	**freeBlock(buf); buf[0] + buf[1] + ... +buf[255] = 0**
5	**freePool()==-1;freePoolForce();** **getBlockCount()==0;getAvailableBlockCount()==0;**