Memory manager

Submission deadline:	**2023-05-15 11:59:59**
Late submission with malus:	**2023-05-19 23:59:59** (Late submission malus: 100.0000 %)
Evaluation:	**12.0000**
Max. assessment:	**30.0000** (Without bonus points)
Submissions:	3 / 60
Advices:	0 / 0

This task is focused on the memory allocation problem that is done by the OS and the runtime of user programs. The problem is to implement a simplified version of a memory manager.

We assume a program where the dynamic memory allocator does not exist (it is disabled in the testing Progtest environment). Thus, there is no `malloc`, `free`, ..., nor C++ operators `new` or `delete`. Your task is to implement variants of the functions.

Your implementation will be given a continuous memory pool of a given size. The memory pool will be sized between 1MiB and several hundreds MiB. Your functions will manage the memory. That is, the rest of the program may ask your function to allocate some block of the memory and subsequently the block may be released. Your implementation must keep track of the allocated/free regions, must support allocation/releasing of a block, and joining of freed adjacent blocks.

The implementation consists of the following functions:

```
int HeapInit ( void * startAddr, int size );
```

> Function `HeapInit` will be invoked first. The parameters are: pointer `startAddr` that points to the beginning of the managed memory pool and `size` which is the size (in bytes) of the managed memory pool.
>
> The passed memory pool will be used to simulate the heap. The subsequent requests shall allocate memory from that pool. Moreover, you shall use the passed memory pool to store your service information (e.g. the information which blocks were allocated/freed).

```
void HeapDone ( int * pendingBlk );
```

> Function `HeapDone`, will be called when the heap is to be destroyed. The function will report a summary on how many blocks were left unfreed in the managed memory pool. The number of unfreed blocks is passed via output parameter `pendingBlk`.

```
void * HeapAlloc ( int size );
```

> Function `HeapAlloc` is used to allocate a memory block on the heap. The use is similar to `malloc`. The parameter is the size of the memory block to allocate (in bytes), the return value is a pointer to the beginning of the allocated contiguous block of size at least `size`. If the function fails (e.g. there is no memory left or all available blocks are smaller than the required size), the return value shall be NULL.

```
bool HeapFree ( void * blk );
```

> Function `HeapFree` is used to free previously allocated memory block. The parameter is a pointer to the memory block as it was returned by `HeapAlloc`. The function returns either true (success), or false if it fails to free the block (the pointer is invalid, block has been already freed, ...).

Submit a source file with the implementation of the four required functions, your auxiliary functions, data types and declarations. Use the example below as a basis for your implementation. If the conditional compile directives are left unchanged, the source can be used to test the program locally as well as the source may be submitted to the Progtest.

Your implementation is significantly limited (see the list of available header files). There is no STL, no C dynamic allocation function (malloc, ...) and no C++ memory allocation operators (new, ...). The program will be executed in a limited testing environment. There is a strict limit on the available memory. In addition to the managed memory pool, there is only a few kilobytes available for your data segment and stack. This extra memory is small, it is not big enough to store the required service information (you must keep the service information inside the managed memory pool).

There are mandatory and optional tests in this homework. The mandatory tests check whether your implementation correctly allocates the memory blocks (blocks do not overlap, freed adjacent blocks are joined, ...).

Optional tests check the handling of fragmentation and the speed of the implementation. If these tests are not passed, some points will be awarded, however, there will be a penalty.

Sample data: **Download**

Reference

- **Evaluator: computer**
  - Program compiled
  - Test 'Zakladni test podle ukazky': success
    - result: 100.00 %, required: 100.00 %
    - Total run time: 0.005 s (limit: 6.000 s)
    - Peak mem usage: 88372 KiB (limit: 88754 KiB)
    - Mandatory test success, evaluation: 100.00 %
  - Test 'Jeden velky blok': success
    - result: 100.00 %, required: 100.00 %
    - Total run time: 0.059 s (limit: 5.995 s)
    - Peak mem usage: 88372 KiB (limit: 88754 KiB)
    - Mandatory test success, evaluation: 100.00 %
  - Test 'Mnoho malych bloku': success
    - result: 100.00 %, required: 100.00 %
    - Total run time: 0.007 s (limit: 5.936 s)
    - Peak mem usage: 88372 KiB (limit: 88754 KiB)
    - Mandatory test success, evaluation: 100.00 %
  - Test 'Nahodne operace': success
    - result: 100.00 %, required: 100.00 %
    - Total run time: 0.052 s (limit: 5.929 s)
    - Peak mem usage: 88372 KiB (limit: 88754 KiB)
    - Mandatory test success, evaluation: 100.00 %
  - Test 'Test fragmentace': success
    - result: 100.00 %, required: 50.00 %
    - Total run time: 0.049 s (limit: 15.000 s)
    - Peak mem usage: 88372 KiB (limit: 88754 KiB)
    - Optional test success, evaluation: 100.00 %
  - Test 'Test rychlosti': success
    - result: 100.00 %, required: 80.00 %
    - Total run time: 0.174 s (limit: 14.951 s)
    - Peak mem usage: 88372 KiB (limit: 88754 KiB)
    - Optional test success, evaluation: 100.00 %
  - Overall ratio: 100.00 % (= 1.00 * 1.00 * 1.00 * 1.00 * 1.00 * 1.00)
- Total percent: 100.00 %
- Early submission bonus: 3.00
- Total points: 1.00 * (30.00 + 3.00) = 33.00

SW metrics:

	Total	Average	Maximum	Function name
Functions:	**14**	--	--	--
Lines of code:	**215**	**15.36 ± 14.18**	**49**	`CMemAllocator::Init`
Cyclomatic complexity:	**41**	**2.93 ± 2.34**	**7**	`CMemAllocator::Malloc`

3	**2023-05-04 15:52:14**	**Download**
Submission status:	Evaluated	
Evaluation:	12.0000	

- **Evaluator: computer**
  - Program compiled
  - Test 'Basic test': success
    - result: 100.00 %, required: 100.00 %
    - Total run time: 0.004 s (limit: 6.000 s)
    - Peak mem usage: 88372 KiB (limit: 88754 KiB)
    - Mandatory test success, evaluation: 100.00 %
  - Test 'One big block': success
    - result: 100.00 %, required: 100.00 %
    - Total run time: 0.056 s (limit: 5.996 s)
    - Peak mem usage: 88372 KiB (limit: 88754 KiB)
    - Mandatory test success, evaluation: 100.00 %
  - Test 'Many small blocks': success
    - result: 100.00 %, required: 100.00 %
    - Total run time: 0.024 s (limit: 5.940 s)
    - Peak mem usage: 88372 KiB (limit: 88754 KiB)
    - Mandatory test success, evaluation: 100.00 %
  - Test 'Random operations': success
    - result: 100.00 %, required: 100.00 %
    - Total run time: 0.300 s (limit: 5.916 s)
    - Peak mem usage: 88372 KiB (limit: 88754 KiB)
    - Mandatory test success, evaluation: 100.00 %
  - Test 'Fragmentation': failed
    - result: 0.00 %, required: 50.00 %
    - Total run time: 2.678 s (limit: 15.000 s)
    - Peak mem usage: 88372 KiB (limit: 88754 KiB)
    - Optional test failed, evaluation: 50.00 %
    - Failed (invalid output)
    - Failed (invalid output)
  - Test 'Speed test': Abnormal program termination (Time limit exceeded)
    - Cumulative test time exceeded, killed after:: 12.332 s (limit: 12.322 s)
    - Optional test failed, evaluation: 80.00 %
  - Overall ratio: 40.00 % (= 1.00 * 1.00 * 1.00 * 1.00 * 0.50 * 0.80)
- Total percent: 40.00 %
- Total points: 0.40 * 30.00 = 12.00

SW metrics:

	Total	Average	Maximum	Function name
Functions:	**5**	--	--	--
Lines of code:	**108**	**21.60 ± 18.86**	**57**	`main`
Cyclomatic complexity:	**14**	**2.80 ± 2.23**	**6**	`HeapFree`

2	**2023-05-04 15:44:55**	**Download**
Submission status:	Evaluated	
Evaluation:	10.8000	

- **Evaluator: computer**
  - Compile in 'pedantic' mode failed (10 % penalty).
  - Test 'Basic test': success
    - result: 100.00 %, required: 100.00 %
    - Total run time: 0.006 s (limit: 6.000 s)
    - Peak mem usage: 88372 KiB (limit: 88754 KiB)
    - Mandatory test success, evaluation: 100.00 %
  - Test 'One big block': success
    - result: 100.00 %, required: 100.00 %
    - Total run time: 0.059 s (limit: 5.994 s)
    - Peak mem usage: 88372 KiB (limit: 88754 KiB)
    - Mandatory test success, evaluation: 100.00 %
  - Test 'Many small blocks': success
    - result: 100.00 %, required: 100.00 %

        - Total run time: 0.023 s (limit: 5.935 s)
        - Peak mem usage: 88372 KiB (limit: 88754 KiB)
        - Mandatory test success, evaluation: 100.00 %
    - Test 'Random operations': success
        - result: 100.00 %, required: 100.00 %
        - Total run time: 0.285 s (limit: 5.912 s)
        - Peak mem usage: 88372 KiB (limit: 88754 KiB)
        - Mandatory test success, evaluation: 100.00 %
    - Test 'Fragmentation': failed
        - result: 0.00 %, required: 50.00 %
        - Total run time: 2.965 s (limit: 15.000 s)
        - Peak mem usage: 88372 KiB (limit: 88754 KiB)
        - Optional test failed, evaluation: 50.00 %
        - Failed (invalid output)
        - Failed (invalid output)
    - Test 'Speed test': Abnormal program termination (Time limit exceeded)
        - Cumulative test time exceeded, killed after:: 12.042 s (limit: 12.035 s)
        - Optional test failed, evaluation: 80.00 %
    - Overall ratio: 36.00 % (= (1.00 * 1.00 * 1.00 * 1.00 * 0.50 * 0.80) * 0.9)
- Total percent: 36.00 %
- Total points: 0.36 * 30.00 = 10.80

SW metrics:

	Total	Average	Maximum	Function name
Functions:	**5**	--	--	--
Lines of code:	**108**	**21.60 ± 18.86**	**57**	`main`
Cyclomatic complexity:	**14**	**2.80 ± 2.23**	**6**	`HeapFree`

1	**2023-05-01 18:45:35**	**Download**
Submission status:	Evaluated	
Evaluation:	12.0000	

- **Evaluator: computer**
    - Program compiled
    - Test 'Basic test': success
        - result: 100.00 %, required: 100.00 %
        - Total run time: 0.006 s (limit: 6.000 s)
        - Peak mem usage: 88372 KiB (limit: 88754 KiB)
        - Mandatory test success, evaluation: 100.00 %
    - Test 'One big block': success
        - result: 100.00 %, required: 100.00 %
        - Total run time: 0.057 s (limit: 5.994 s)
        - Peak mem usage: 88372 KiB (limit: 88754 KiB)
        - Mandatory test success, evaluation: 100.00 %
    - Test 'Many small blocks': success
        - result: 100.00 %, required: 100.00 %
        - Total run time: 0.024 s (limit: 5.937 s)
        - Peak mem usage: 88372 KiB (limit: 88754 KiB)
        - Mandatory test success, evaluation: 100.00 %
    - Test 'Random operations': success
        - result: 100.00 %, required: 100.00 %
        - Total run time: 0.332 s (limit: 5.913 s)
        - Peak mem usage: 88372 KiB (limit: 88754 KiB)
        - Mandatory test success, evaluation: 100.00 %
    - Test 'Fragmentation': failed
        - result: 0.00 %, required: 50.00 %
        - Total run time: 3.284 s (limit: 15.000 s)
        - Peak mem usage: 88372 KiB (limit: 88754 KiB)
        - Optional test failed, evaluation: 50.00 %
        - Failed (invalid output)
        - Failed (invalid output)

- Test 'Speed test': Abnormal program termination (Time limit exceeded)
  - Cumulative test time exceeded, killed after:: 11.723 s (limit: 11.716 s)
  - Optional test failed, evaluation: 80.00 %
- Overall ratio: 40.00 % (= 1.00 * 1.00 * 1.00 * 1.00 * 0.50 * 0.80)

- Total percent: 40.00 %
- Total points: 0.40 * 30.00 = 12.00

SW metrics:

	Total	Average	Maximum	Function name
Functions:	**5**	--	--	--
Lines of code:	**109**	**21.80 ± 18.19**	**57**	`main`
Cyclomatic complexity:	**15**	**3.00 ± 1.79**	**5**	`HeapAlloc`