Practice #4 (for lecture note #4)

Memory Management

Due: 9pm, Friday, 2019-10-25

Notes and Advices:

1. Submission of the practice is NOT mandatory. The main purpose of these problems is for you to practice and get refreshed on the contents of the lecture note for the coming class meeting. Practice of the problems before class is HIGHLY recommended.
2. However, in case you need to "bargain" your term grade while you are below the next/higher letter grade for a small enough margin, you can talk to me if you have been consistently submitting your practices in time.
3. Note that your submission of the practice will NOT be graded or checked (by me or TA). Again, this is for your practice only.
4. The submissions on NTU CEIBA have strict deadline. Usually it is before midnight of the class meeting.
5. Practice problems will be, hopefully, uploaded to CEIBA no later than the previous weekend of the class.

Problems // "(pn)" refers to the page #n in the lecture note

1. (p6) Try the following code:

```
int main()
{
   int a[4] = {0};
   int b = 100;
   cout << &a << endl;
   cout << &b << endl;
   // HERE…
   cout << b << endl;
}
```

   ==> Insert code to "HERE…"
   - Try to assign to an out-of-bound array cell of '`a`' that collides with the memory address of '`b`'. What would be the value of '`b`'?
   - Try to assign to an out-of-bound array cell of '`a`' that does not belong to any variable. Any error?

2. (p8) Try the code in p7 of lecture note #4.
   - Explain the outputs.
   - At the end, add the following:

```
*p = 40;
cout << k << endl;
cout << *q << endl;
```

     ==> What is the output?
   - What if we assign "p = 0" right after it is deleted?

3. (p27) Try the code in p26 of lecture note #4.
   - What's the runtime difference? Try to change the array size (e.g. 1 << 22) and run again.
   - Count how many times A's constructor is called.

- Does the speedup come from the reduction of constructor calls? What is the runtime percentage from "`*(a[i]) = i`"?

4. (p40) Define an array `int a[10];`
   - Can you figure out where the size of array (i.e. 10) is stored?
   - Change it to "`int *a = new int[10]`". Any difference?

   Define a `class A` and an array `A a[10]`
   - Can you figure out where the size of array (i.e. 10) is stored?
   - Change it to "`A *a = new A[10]`". Any difference?

   Add a destructor for `class A`. Any change?

5. (p49) Let's practice a very simple memory manager
   - Define a `class Mem`, with three data members: `char* _begin, _ptr, _end`; as explained in p48 of lecture note #4. Its constructor takes a `size_t` (default = 65536) as the number of Bytes of managed memory.
   - Define a member function "`A* Mem::alloc(size_t t)`" that allocates memory from managed memory for `t` Bytes.
   - Define a `class A`, with arbitrary data members and whose memory is managed by `class Mem`. Overload its "`new`" operator.
   - In `main()`, define some `A*` variables and print out message properly to verify that the memory is well managed.

   What improvements you can think of from this simple example?

6. (p56) Design a simple recycling list
   - Define a `class A` of size = 24 Bytes.
   - Define a global variable "`A *aa = new A[SIZE];`", where `SIZE = 1024`
   - In `main()`, randomly generate 10 numbers between [0, SIZE). Recycle these elements by connecting their memory as described in the previous page. That is, first-in-last-out.
   - Try to retrieve memory from the lastly recycled one. Verify their indices are correct.

7. (p74) Complete the example in the previous slide. For each exception object, try to define a member function to print out some message to make sure they are properly caught.