## Dynamic Memory Management

We must keep track of the heap structure for memory management. Maximum size of saving data is 1KB, and malloc and free uses virtual memory address as an argument. Virtual address is decided by adding available offset in virtual memory table `self.table2` and 0x100000 which is `base\_address` of VM. Allocated virtual address of each variables never change until free command occurs.

Address of the variable and allocating size will be saved in `self.dic` in heap structure. In heap structure, some important functions needed to be declared as sub-functions of the heap for memory management which is coded in `memory.py` is as below:

```

class Heap:

malloc (self, size):

...

free (self, var):

...

mem (self):

...

dynamic\_allocation (self, size):

...

```

\* `malloc`: The function call `malloc(size)` in the input code invokes a call to `x = heap.malloc(size)` method at Runtime. The method to find possible n size of memory space for variable x. `self.table2` is used for getting virtual address for allocated space. If malloc successes, it returns the virtual address.

Two possible errors might occur: the `size` might be a negative value, or there might be not enough space left in the heap. If there is no enough space for such size, execute defragmentation in the memory space. If the defragmentation process fails or there is still not enough memory left, it returns 0(which will print “Out of memory”). If `size` is a negative value, it returns 0(which will print “size\_t should be a positive integer”).

\* `free`: The function call `free(var)` in the input code invokes a call to `heap.free(var)` method at Runtime. The method will deallocate memory space corresponding to `var` which is the address of such variable. If there is no allocated space for address `var`, it returns 0(which will print “not allocated address”).

\* `mem`, The command line `mem` invokes a call to `heap.mem()` method in runtime. This will print out the message “Dynamic allocation : x, y” where x is the number of allocated variables and y is the total memory size of the allocated space.

\* ` dynamic\_allocation`, The function call `self.dynamic\_allocation(self, size)` in the code `memory.py` is called to defragment the memory space when there is not enough space left for size `size` in the memory. When defragmentation occurs, the data moves forward and it updates `self.dic` for changed offset `off`. The table `self.table` moves with it too.

If this function is called, it prints “defragmentation operated”.