# CSC45500 Programming Project #3
# Due: Thursday, March 21, 11:59PM

## Objectives

- implementing a programming language memory manager
- implement a garbage collection system

## Problem Statement

You are building an interpreter that supports allocating dynamic memory to variables, copying variables (references), free allocated space, garbage collection, and freelist compression. This will be accomplished through a simple programming language interpreted through a grammar based recursive descent parser and a lexical analyzer.

## The Lexical Analyzer and Parser

The lexical analyzer should return tokens of the same types as found in project 1.

The parser should accept an input sequence matching the following BNF grammar, with `<prog>` being the starting rule:

```
<prog> → <slist>
<slist> → <stmt> SEMICOLON <slist> | ϵ
<stmt> → ID LPAREN ID RPAREN |
          ID LPAREN RPAREN |
          ID ASSIGNOP <rhs>
<rhs> → ID LPAREN NUM_INT RPAREN |
          ID
```

This grammar accepts a sequence of statements, each of which is followed by a ';' . Each statement is one of the following options:

- **`free(variableName)`** which should return any storage associated with `variableName` to the freeList of available blocks. ( <stmt> → **ID LPAREN ID RPAREN** )
- **`dump()`** which should print each currently allocated variable, it's location in memory, the size of its allocation, and the associated memory block's reference count. After printing out each variable currently associated with a memory location, this should print each block in the feeeList (start location and size, noting that anything in the freeList should have a reference count of 0.) ( <stmt> → **ID LPAREN RPAREN** )
- **`compress()`** which should join adjacent blocks in the freeLIst into an appropriately located and sized larger bock. Non-adjacent blocks should be left alone. ( <stmt> → **ID LPAREN RPAREN** )
- **`variableName = alloc(integerAmount)`** which should use *first fit* in the freeList to allocate the requested integerAmount of space and allocate it to the variableName specified. (*combination of* <stmt> → **ID ASSIGN** <rhs> *and* <rhs> → **ID LPAREN NUM_INT RPAREN** )
- **`variableName = otherVariableName`** which should take the reference found associated with otherVariableName and also associate it with variableName. (*combination of* <stmt> → **ID ASSIGN** <rhs> *and* <rhs> → **ID**)

Note that the freeList should always remain sorted from lowest address free block to highest address free block. Compression (a.k.a. compaction) should only be performed by the user input program calling compress(); … *do not automatically perform compression/compaction without an explicit call to do so.*

## Problem Statement

You are to write a program that:

1. prompt the user to enter the initial (single) block size for the freeList.
2. prompt the user to enter the name of an input file. This input file will match the grammar above (you are guaranteed there will be no syntax errors in the input file!)
3. Processes the input file according to the rules specified in the previous section.

## Example Execution

Suppose you have the following input file (`test.myl`):

```
c = alloc(34 ); dump();
c=alloc(17); dump();
b = alloc(57); dump();
a = alloc(3); dump();
d=a; dump();
free(a); dump();
free(b); dump();
free(c); dump();
d=d; dump();
hello=alloc(52); dump();
compress(); dump(); free(d); compress(); dump();
goodbye=hello; bonjour=hello;
hola=bonjour; privet=hola; dump();
free(hello); dump();
```

The following is an example execution of the program using the above input file. The `output from the program is in regular text`, but the ***`user input is in italics`*** … you do not need (nor should you try) to get your program to match this pattern.

```
Please enter the initial freelist (heap) size: 512
Please enter the name of an input file: test.myl
Variables:
c:0(34) [1]
Free List:
34(478) [0]
```

address of c

c's reference count

size of c's allocation

c = alloc(34); dump();

```
================================================================
Variables:
c:0(17) [1]
Free List:
17(17) [0], 34(478) [0]
================================================================
```

c = alloc(17); dump();

```
Variables:
b:34(57) [1]
c:0(17) [1]
Free List:                          b = alloc(57); dump();
17(17) [0], 91(421) [0]
================================================================
Variables:
a:17(3) [1]
b:34(57) [1]                        a = alloc(3); dump();
c:0(17) [1]
Free List:
20(14) [0], 91(421) [0]
================================================================
Variables:
a:17(3) [2]
b:34(57) [1]                            d = a; dump();
c:0(17) [1]
d:17(3) [2]
Free List:
20(14) [0], 91(421) [0]
================================================================
Variables:
b:34(57) [1]                           free(a); dump();
c:0(17) [1]
d:17(3) [1]
Free List:
20(14) [0], 91(421) [0]
================================================================
Variables:
c:0(17) [1]                              free(b); dump();
d:17(3) [1]
Free List:
20(14) [0], 34(57) [0], 91(421) [0]
================================================================
Variables:
d:17(3) [1]                            free(c); dump();
Free List:
0(17) [0], 20(14) [0], 34(57) [0], 91(421) [0]
================================================================
Variables:
d:17(3) [1]                              d=d; dump();
Free List:
0(17) [0], 20(14) [0], 34(57) [0], 91(421) [0]
================================================================
```

```
Variables:
d:17(3) [1]
hello:34(52) [1]          hello=alloc(52); dump();
Free List:
0(17) [0], 20(14) [0], 86(5) [0], 91(421) [0]
================================================================
```

*(space left here for a reference point in the input - the compress calls.)*

```
Variables:
d:17(3) [1]                    compress(); dump();
hello:34(52) [1]
Free List:
0(17) [0], 20(14) [0], 86(426) [0]
================================================================
Variables:
hello:34(52) [1]
Free List:                          free(d); compress(); dump();
0(34) [0], 86(426) [0]
================================================================
Variables:
bonjour:34(52) [5]                        goodbye=hello;
goodbye:34(52) [5]                  bonjour=hello;hola=bonjour;
hello:34(52) [5]                       privet=hola; dump();
hola:34(52) [5]
privet:34(52) [5]
Free List:
0(34) [0], 86(426) [0]
================================================================
Variables:
bonjour:34(52) [4]
goodbye:34(52) [4]                        free(hello); dump();
hola:34(52) [4]
privet:34(52) [4]
Free List:
0(34) [0], 86(426) [0]
================================================================
```

Note that this test is nowhere near exhaustive. *It is 100% up to you to come up with more exhaustive tests than the above!*

## What To Hand In

You will be submitting a zip or tgz file containing your source code (.cpp and .h files) and a `read.me` file to Canvas. Make sure that you place the project into a single folder (which may contain sub-folders).

The `read.me` file should include information about your project including (but not limited to):
• your name
• the date
• the platform you developed your code on (Windows, Linux, …)
• any special steps needed to compile your project
• any bugs your program has
• a brief summary of how you approached the problem
You might also want to consider adding things like a “software engineering log” or anything else you utilized while completing the project.

## Grading Breakdown

| | |
|---|---|
| Correct Submission | 10% |
| Code Compiles | 20% |
| Following Directions | 20% |
| Correct Execution | 40% |
| Code Formatting/Comments/read.me | 10% |
| *Early Submission Bonus* | 5% |

## Final Notes & Warnings

- This is not the kind of project you can start the night it is due and complete on time. My recommendation is to start *now.*
- *Start now!*
- Projects may *not* be worked on in groups or be copied (either in whole or in part) from *anyone* or *ANYWHERE*. Failure to abide by this policy *WILL* result in disciplinary action(s). See the course syllabus for details.
- *Have you started working on this project yet? If not, then* ***START NOW!!!!***