Project Information:

Our project name is Heap Heap Hooray, and our project is the development of a garbage collector for the MiniJava compiler. This project requires us to deep-dive into garbage collector implementations, understanding and learning from them to write one that caters to the project's goals.

Our team consists of Tyler Gutowski (tgutowski2020@my.fit.edu), and Trevor Schiff (tschiff2020@my.fit.edu)**.** Our advisor and client is Dr. Ryan Stansifer (ryan@fit.edu), a compiler researcher who has a deep understanding of the MiniJava compiler and Java runtime.

Our primary goal for Milestone 2 was to implement the reference counting algorithm, and our secondary goal was to implement mark-and-sweep. We completed our primary goal, but did not begin working on our second goal. The mark-and-sweep algorithm has become our primary goal for Milestone 3.

Progress Matrix:

Task	**Progress**	**Tyler**	**Trevor**
1. Heap	100%	0.0	1.0
2. Allocator	100%	0.2	0.8
3. Reference counting initialization handling	100%	0.5	0.5
4. Reference counting reference handling	100%	0.5	0.5

5. Reference counting argument handling	100%	0.5	0.5
6. Environment Setup (SPARC, Jabberwocky, QEMU)	100%	1.0	0.0
7. Memory Tests	100%	0.5	0.5
8. Mark-and-Sweep implementation	0%	0.0	0.0

Discussion:

1. Add HeapHeader to store information about the Object. This information helps the GC keep necessary information regarding reference counting and marks for mark-and-sweep.
2. Added methods for heap allocation management (allocating, freeing, checking addresses) and functions for incrementing and decrementing reference counting of the heap allocations.
3. Added functionality to ensure an object is initialized with a reference counter, via heap_alloc(), which initializes the reference counter at 0, which increments upon reference.
4. The reference counter gets incremented when a reference is passed.
5. The reference counter gets incremented when an object is passed in as a function argument. This happens in the prologue of the function, where the counter is incremented, and in the epilogue it decrements.
6. We set up SPARC on a local machine to run the MiniJava compiler. We chose local over Andrew because we might need specific permissions at a later date. The project uses the containerization software Jabberwocky (https://github.com/Kippiii/jabberwocky-container-manager) from a previous senior project. It was written by Ian Orzel and Dylan McDougall. We chose this software for our containerization because there are pre-made containers for C, C++, Ada, Go, Rust, Fortran, Haskell and Prolog for Programming Language Concepts, OS/161 for Operating Systems, and the SPARC architecture for Compiler Theory. Since the compiler we are

using is from the Compiler Theory class, it was simple. Setting up the environments was very difficult for multiple reasons. First, I was hoping to use my clustered Raspberry Pi server to run SPARC through QEMU. This proved to be difficult, because the SPARC installation kept crashing due to the limited resources (Debian 11 running a SPARC emulator). Then we decided that it would be beneficial to just use Jabberwocky, which runs a Debian container with QEMU that runs SPARC, but my Raspberry Pi server was overloaded. I tried running it natively on Windows, but because the Makefiles were written for Linux, I installed Ubuntu WSL and worked from there.

7. We ran a few simple MiniJava tests where we initialized new objects, then added and removed a few references to ensure the reference counter worked. We also ran a clobbering test where we overwrote an already existing object with the expectation that its counter would decrement.
8. The Mark-and-sweep algorithm was a secondary task that didn't get completed. It will become the primary task for our Milestone 3.

Contributions:

Tyler Gutowski: Helped in writing the allocator, and half of the reference counting software. The reference counting software keeps track of when objects are initialized, referenced, and passed in as function arguments. Tyler also set up the Jabberwocky container and ran a SPARC environment locally to run the garbage collector with tests he wrote.

Trevor Schiff: Designed and wrote the HeapHeader from scratch, which allows the GC to grab the necessary information about objects. Also wrote the allocator and deallocator, which helps the reference counter keep track of which objects are still in use. Wrote most of the reference counting software to determine when objects were initialized, referenced, and passed in as function arguments. Trevor helped in writing the MiniJava test files which helped to show how the objects were being referenced.

Next Milestone:

Task	Tyler	Trevor
Implementation of the Mark Phase algorithm	0.5	0.5
Implementation of the Sweep Phase algorithm	0.5	0.5
Mark-and-Sweep testing	0.8	0.2
Dividing the heap into multiple parts for defragmentation	0.2	0.8

Implementation of the Copying Algorithm	0.5	0.5
Copying Algorithm testing	0.5	0.5
Integration with Reference Counting	0.2	0.8
Comparing metrics between different GC algorithms	0.5	0.5

Discussion:
In our next milestone, we're diving deep into the Mark-and-Sweep and Copying garbage collection algorithms. We're both going to be tackling the Mark Phase and the Sweep Phase. We understand how important it is to get these right. Trevor will be taking a bit more of the load during the heap division for the Copying Algorithm, due to his experience with compilers, while Tyler will work on the testing in more depth. We'll both be hands-on when implementing and testing the Copying Algorithm, because it should be similar to the Mark-and-Sweep algorithm. Our goal is to wrap things up by smoothly integrating everything with the existing reference counting system, and comparing the metrics.

Client Meetings:

See Faculty Advisor Meetings below

Client Feedback:

See Faculty Advisor Feedback below

Faculty Advisor Meetings:

We had one meeting with Dr. Stansifer on October 23rd to discuss our ideas with the environment, and another meeting on October 30th to show our progress and to give a short demonstration.

Faculty Advisor Feedback:

In email.

Approval from Faculty Advisor:

"I have discussed with the team and approve this project plan. I will evaluate the progress and assign a grade for each of the three milestones."

Signature: _____ Date: _____

Evaluation by Faculty Advisor:

Tyler Gutowski	0	1	2	3	4	5	5.5	6	6.5	7	7.5	8	8.5	9	9.5	10
Trevor Schiff	0	1	2	3	4	5	5.5	6	6.5	7	7.5	8	8.5	9	9.5	10

Signature: _____ Date: _____