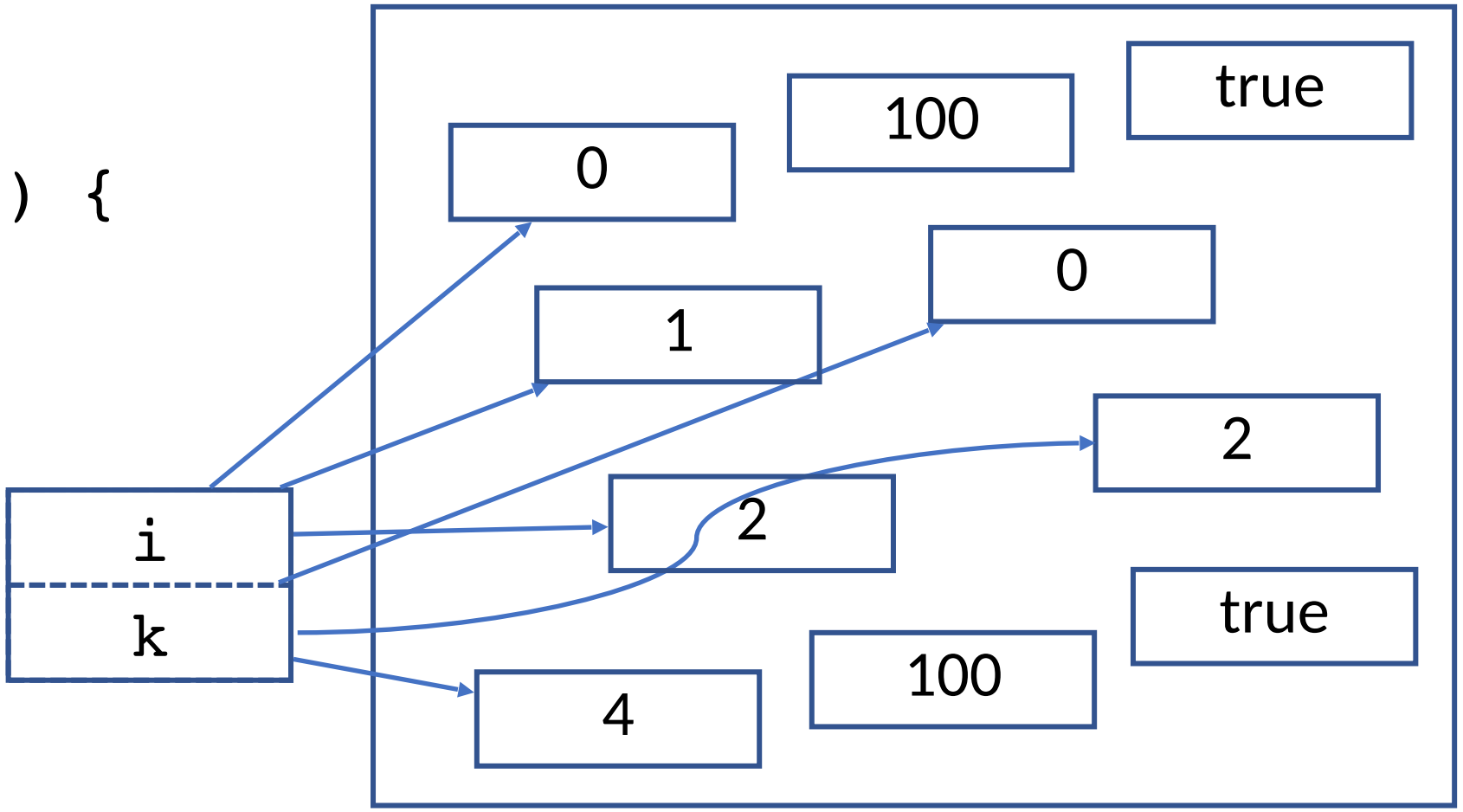
Phase 3 Overview

6.112

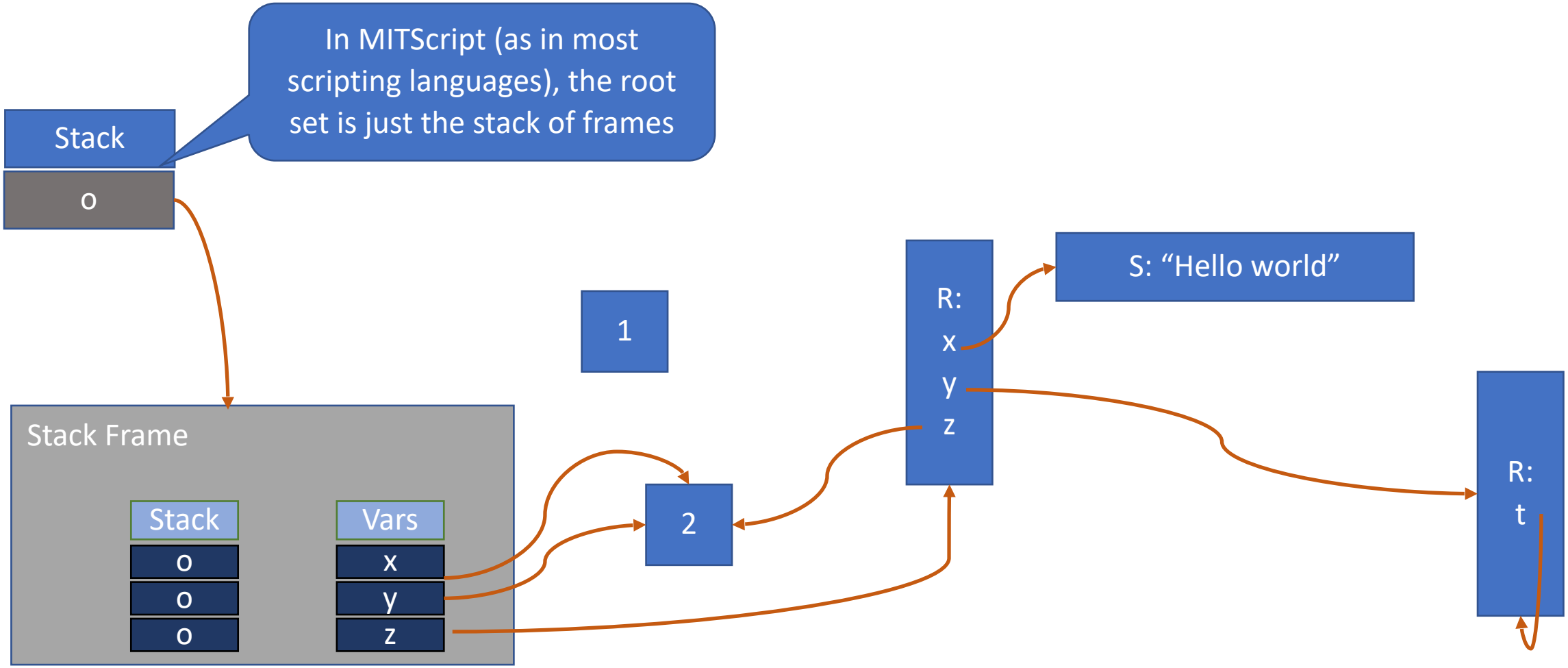
Fall 2025

Phase 3: Garbage Collection

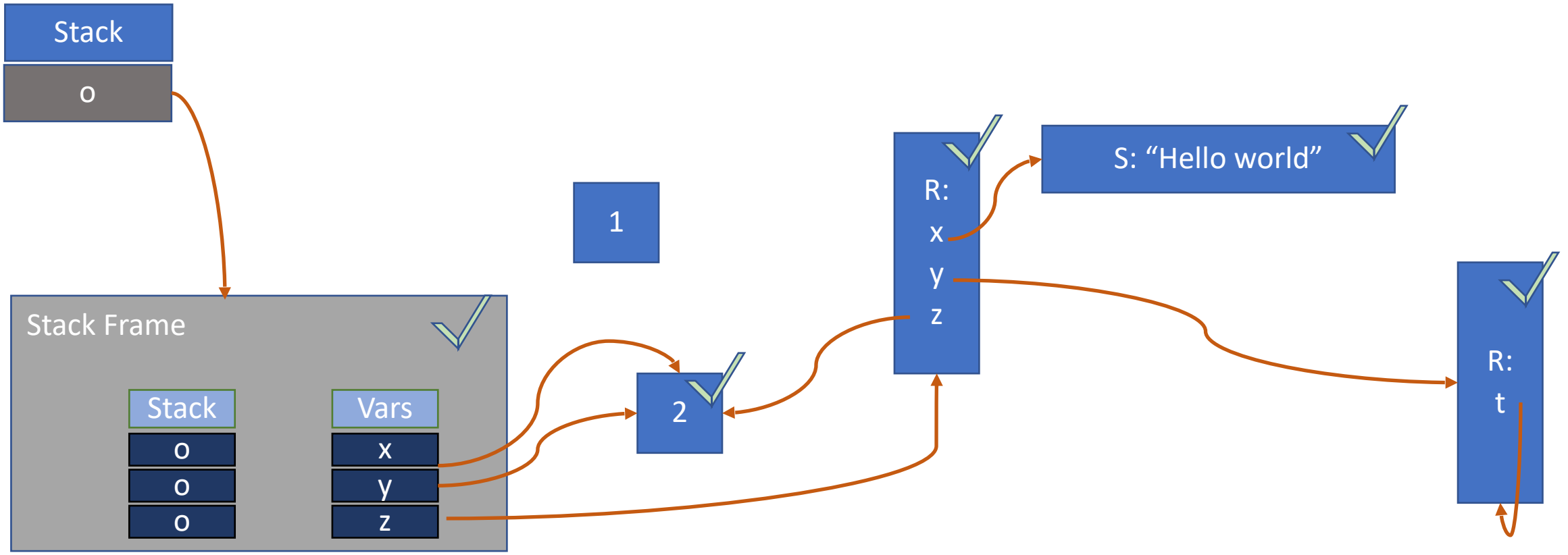
```
i = 0;
k = 0;
while (i < 100) {
   k = k + 2;
   i = i + 1;
}
```



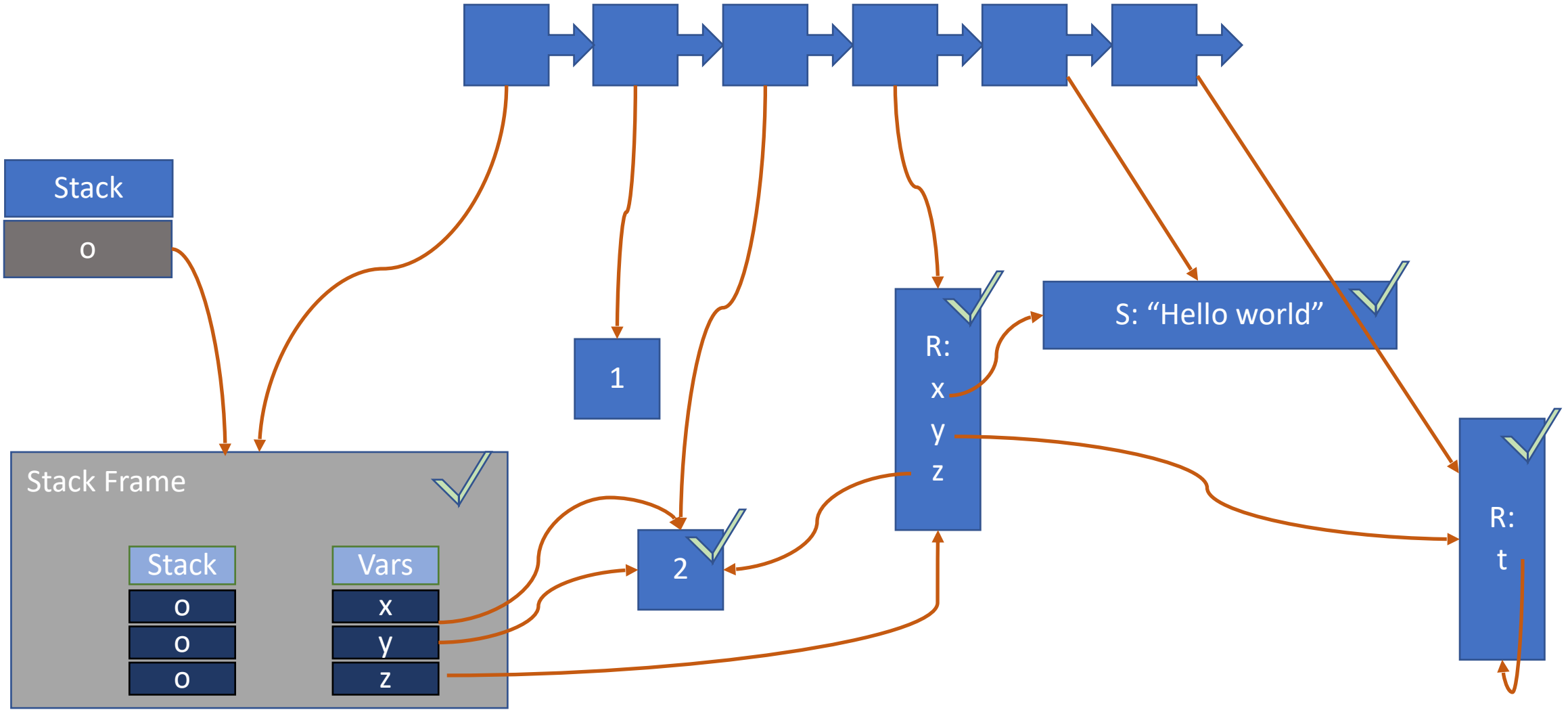
Mark-and-Sweep Algorithm



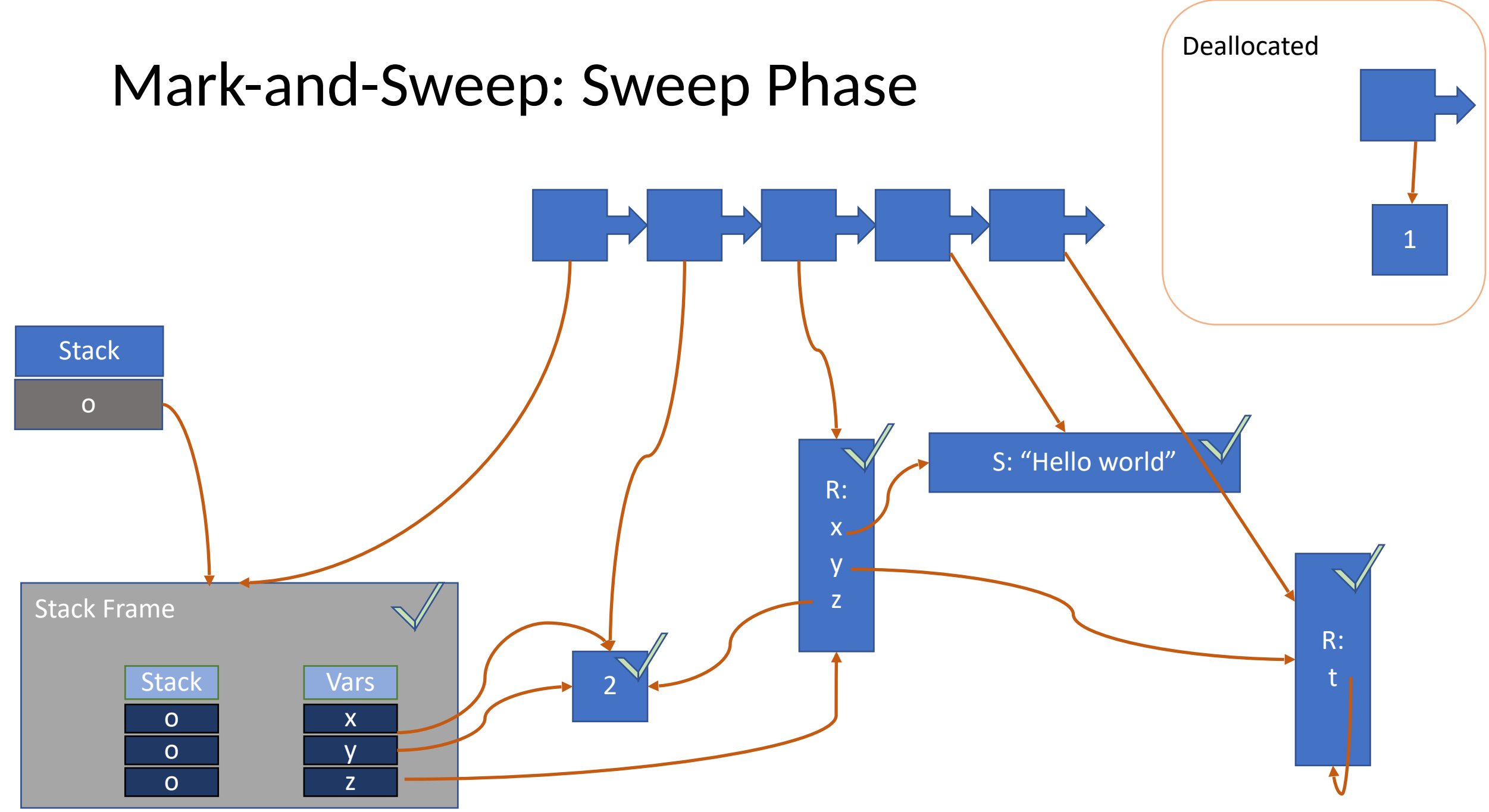
Mark-and-Sweep: Mark Phase



Mark-and-Sweep: Sweep Phase



Mark-and-Sweep: Sweep Phase



Garbage Collector Interface

```
class CollectedHeap {
  template<typename T, typename ...Args>
  T* allocate(Args&& ...args);

  template <typename I> void gc(I begin, I end);
  void markSuccessors(Collectable* obj);
};
```

Garbage Collector Interface

```
class Collectable {
  virtual void follow(CollectedHeap& heap) = 0;
};
```

Deliverables (Due October 17@ 10 pm ET)

- Your mark-and-sweep garbage collector, implemented as a header-only library in src/gc/gc.hpp
  - Interface must conform to what is provided in the skeleton
  - Make sure your implementation works with provided C++ test
- Report
- Five test cases

Tips for Optimizing Your P3 Deliverables

- Measure memory usage carefully
  - It's easy to miss an automatic allocation
  - Need to carefully estimate memory usage of STL data structures (or use custom replacements)
  - Use Valgrind (memcheck and massif) to profile and debug
- Reuse value objects (such as Booleans and None)
- Compile your VM with `-O2` (or `-O3`) flag
  - Enabled by default in the Release profile in CMake
- Focus on correctness before optimizing!
- Always measure performance before optimizing