1:

```
→  inet3101-dynamic-memory git:(main) ✗ gcc 1-usestaticarray.c -o staticarray
→  inet3101-dynamic-memory git:(main) ✗ ./staticarray



Index 0 of the array1 starts at: 0x7ff7b46a61d0
Index 1 of the array2 starts at: 0x7ff7b46a61e0

Please enter a number: 5


Current numbers in array2 are:  5  1  1461772368  32760

Current numbers in array3 are:  10  11  12  13

Please enter a number: 4


Current numbers in array2 are:  5  4  1461772368  32760

Current numbers in array3 are:  10  11  12  13

Please enter a number: 3


Current numbers in array2 are:  5  4  3  32760

Current numbers in array3 are:  10  11  12  13

Please enter a number: 6
```

2. N/A

3. In my program I dynamically allocate an array using malloc() based on the user's input. The user can continuously enter numbers and if they exceed the initial allocation, realloc() increases the array size. The program doubles the capacity when needed preventing out-of-bounds errors and allowing unlimited inputs. It also checks if realloc() fails to ensure the program exits safely. This approach efficiently manages memory by starting small and expanding dynamically as needed.

4.

```
→  inet3101-dynamic-memory git:(main) ✗ gcc 4-simple-dynamic-example.c -o simple
→  inet3101-dynamic-memory git:(main) ✗ ./simple


  48

  Variable 'students' starts at 0x7fd742f05f50

  Value at index 0 of allocated memory is: 2

  Size of data at first index is:  4
```

5. An object in Object-Oriented Programming has both data and functions that work with that data. It's like a bundle that groups related things together. A Python list is an object because it doesn't just store numbers or words but also comes with built in functions like .append(), .insert(), and .sort(). These let you change the list without having to manually manage memory like in C. Instead of needing to resize an array yourself, Python takes care of it automatically when you add or remove items. This makes lists more flexible and easier to work with.

6. Using dynamic memory lets us create flexible data structures without needing to know their size ahead of time. A linked list solves a similar problem by allowing us to add or remove elements without worrying about resizing or shifting memory like we would with an array. Instead of storing everything in a fixed block each item in a linked list is stored separately with pointers connecting them. This avoids the need to reallocate memory, but it also means we use more memory for pointers and may have slower access times since we can't just jump to an index like we can in an array.

Headers:

```
● →  header-examples git:(main) ✗ gcc 8-useofheaders.c helloworl
  d.c -o headerprogram
● →  header-examples git:(main) ✗ ./headerprogram
  Hello, World!
```