1.

Output:

```
12

10

5

Segmentation fault
```

This probably (hopefully) isn't what we want to happen. When a class doesn't have a copy constructor explicitly implemented, the compiler will provide an implicit copy constructor. The problem here is that the implicit copy constructor performs a SHALLOW copy rather than a deep copy. This means that the pointer to the array, rather than the array itself is being copied into A2, and later A1, creating a shared reference situation (and an inevitable double delete when their destructors are called at the end of the function call).

2.

a. log(n)

b. nlog(n)

c. n

d. n^2

e. n^2

f. n

3. delete deallocates the item at a single address. delete[] will deallocate all of the items in an array.

4.

Copy: O(n)

Move: O(1)

5.

a. O(n)

b. O(1)

6.

a. 3

b. 2

c. 3

d. 1

7.

a. lval

b. lval

c. rval

d. lval

8.

This function returns a string (name) by reference. However, the string we are returning goes out of scope when we exit the function, so the reference we return is no longer valid. Trying to access it would result in a segmentation fault.