

Sorting, Hashing, and Review: Solutions

1. Selection Sort

1. $\{-4, 17, 3, 94, 46, 8, 29, 12\}$
2. $\{-4, 3, 17, 94, 46, 8, 29, 12\}$
3. $\{-4, 3, 8, 94, 46, 17, 29, 12\}$

2. Merge Sort

1. $\{29, 17, 3, 94\} \{46, 8, -4, 12\}$
2. $\{29, 17\} \{3, 94\} \{46, 8\} \{-4, 12\}$
3. $\{29\} \{17\} \{3\} \{94\} \{46\} \{8\} \{-4\} \{12\}$
4. $\{17, 29\} \{3, 94\} \{8, 46\} \{-4, 12\}$
5. $\{3, 17, 29, 94\} \{-4, 8, 12, 46\}$
6. $\{-4, 3, 8, 12, 17, 29, 46, 94\}$

3. Hashing

1. This is a valid hash function, but every string will get hashed into the same bucket, causing $O(n)$ lookup time.
2. This is not valid because hashing the same string multiple times will result in different codes.
3. This hash function is better than the first, but will not distribute words evenly. Consider words like "eat" and "ate" – they get hashed to the same bucket, even though they are different words.

4. Swap Children (Trees)

```
void swapChildrenAtLevelHelper(BinaryTreeNode* node, int k, int currentLevel) {  
    if (node != nullptr) {  
        if (currentLevel == k) {  
            BinaryTreeNode* temp = node->left;  
            node->left = node->right;  
            node->right = temp;  
        }  
    }  
}
```

```

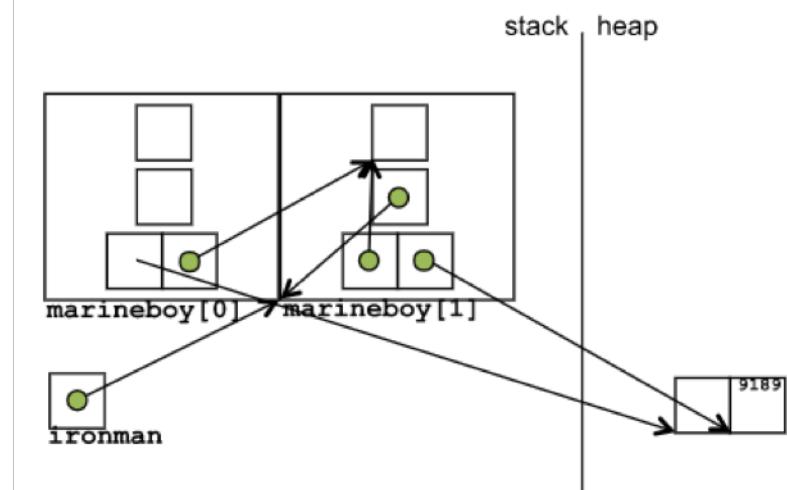
    } else if (currentLevel < k) {
        swapChildrenAtLevelHelper(node->left,  k,
        currentLevel + 1);
        swapChildrenAtLevelHelper(node->right, k, currentLevel + 1);
    }
}

void swapChildrenAtLevel(BinaryTreeNode*& node, int k) {
    if (k <= 0) {
        throw k;
    }
    swapChildrenAtLevelHelper(node, k, 1);
}

```

5. Pointer Trace

State of memory just prior to the call to **barbarella**:



State of memory just before the call to **barbarella** exits:

