**Problem Set 6, Part I**

**Problem 1: Sorting practice**

**1-1)**

first pass: {3, 18, 4, 24, 33, 40, 8, 10, 12}

second pass: **{3, 4, 18, 24, 33, 40, 8, 10, 12}**

**1-2)**

first iteration: {4, 10, 18, 24, 33, 40, 8, 3, 12}

second iteration: {4, 10, 18, 24, 33, 40, 8, 3, 12}

third iteration: {4, 10, 18, 24, 33, 40, 8, 3, 12}

forth iteration: **{4, 10, 18, 24, 33, 40, 8, 3, 12}**

**1-3)**

first pass:

{10, 4, 18, 24, 33, 40, 8, 3, 12}

{10, 4, 18, 24, 33, 8, 40, 3, 12}

{10, 4, 18, 24, 33, 8, 3, 40, 12}

**{10, 4, 18, 24, 33, 8, 3, 12, 40}**

second pass:

{4, 10, 18, 24, 33, 8, 3, 12, 40}

{4, 10, 18, 24, 8, 33, 3, 12, 40}

{4, 10, 18, 24, 8, 3, 33, 12, 40}

**{4, 10, 18, 24, 8, 3, 12, 33, 40}**

third pass:

{4, 10, 18, 24, 8, 3, 12, 33, 40}

{4, 10, 18, 8, 24, 3, 12, 33, 40}

{4, 10, 18, 8, 3, 24, 12, 33, 40}

**{4, 10, 18, 8, 3, 12, 24, 33, 40}**

**1-4)**

pivot = 33

first call:

{10, 18, 4, 24, 33, 40, 8, 3, 12}

swap 33 and 12

{10, 18, 4, 24, 12, 40, 8, 3, 33}

swap 40 and 3

**{10, 18, 4, 24, 12, 3, 8, 40, 33}**

end partition

**1-5)**

second call:

left subarray = {10, 18, 4, 24, 12, 3, 8}

pivot = 24

{10, 18, 4, 24, 12, 3, 8}

swap 24 and 8

{10, 18, 4, 8, 12, 3, 24}

end partition

right subarray = {40, 33}

**{10, 18, 4, 8, 12, 3, 24, 40, 33}**

**1-6)**

first call: {10, 18, 4, 24, 33, 40, 8, 3, 12}

second call: {4, 10, 18, 24, 33, 40, 8, 3, 12}

third call: {4, 10, 18, 24, 33, 40, 8, 3, 12}

forth call: **{4, 10, 18, 24, 33, 40, 8, 3, 12}**

**Problem 2: More Practice with big-O**

**2-1)**

outer loop: 3 times

middle loop: n times

inner loop: (n - 1)/2 times

3 \* n \* (n - 1)/2 = 3(n^2 – n)/2 = **O(n^2)**

**2-2)**

outer loop: n times

inner loop: log n times

n \* log n = **O(n log n)**

**Problem 3: Comparing two algorithms**

*worst-case time efficiency of algorithm A:*

*explanation:*

outer loop: n times

inner loop: (n – 1)/2 times

n \* (n-1)/2 = **O(n^2)**

*worst-case time efficiency of algorithm B:*

*explanation:*

mergesort = O(n log n)

loop: n – 1 times

n – 1 = O(n)

O(n log n) + O(n) = O(n log n + n) = **O(n log n)**

**Problem 4: More practice with sorting - Quicksort**

**4-1)**

pivot = 13

first call:

{14, 7, 27, 13, 24, 20, 10, 33}

swap 14 and 10

{10, 7, 27, 13, 24, 20, 14, 33}

swap 27 and 13

**{10, 7, 13, 27, 24, 20, 14, 33}**

**4-2)**

second call:

left subarray = {10, 7, 13}

pivot = 7

swap 10 and 7

{7, 10, 13}

right subarray = {27, 24, 20, 14, 33}

**{7, 10, 13, 27, 24, 20, 14, 33}**

**Problem 5: More practice with sorting - Mergesort**

**5-1)**

first call: {3, 24, 27, 13, 34, 2, 50, 12}

second call: **{3, 24, 13, 27, 34, 2, 50, 12}**

**5-2)**

third call: {3, 13, 24, 27, 34, 2, 50, 12}

forth call: **{3, 13, 24, 27, 2, 34, 50, 12}**

**5-3)**

the array {24, 3, 27, 13, 34, 2, 50, 12} will be split into two part: left subarray {24, 3, 27, 13} and right subarray {34, 2, 50, 12}

**Problem 6: Practice with references**

**6-1)**

|  |  |  |
| --- | --- | --- |
| **Expression** | **Address** | **Value** |
| n | 0x100 | 0x712 |
| n.ch | 0x712 | 'n' |
| n.prev | 0x718 | 0x064 |
| n.prev.prev | 0x024 | 0x360 |
| n.prev.next.next | 0x722 | null |
| n.prev.prev.next | 0x726 | 0x064 |

**6-2)**

x.next = n;

x.prev = n.prev;

n.prev = x;