CPSC 221 MT1

b0o2b

TOTAL POINTS

44.5 / 63

QUESTION 1

1 CSID **1 / 1**

✓ + **1 pts** **CSID provided**

+ **0 pts** No CSID provided

QUESTION 2

Blizzard 14 pts

2.1 MISC1.1 push pointer **0.5 / 0.5**

✓ + **0.5 pts** **head**

+ **0 pts** tail

+ **0 pts** either

+ **0 pts** No answer, or multiple selections

2.2 MISC1.1 push complexity **0.5 / 0.5**

✓ + **0.5 pts** **$$O(1)$$**

+ **0 pts** $$O(n)$$

+ **0 pts** No answer, or multiple selections

2.3 MISC1.1 pop pointer **0.5 / 0.5**

✓ + **0.5 pts** **head**

+ **0 pts** tail

+ **0 pts** either

+ **0 pts** No answer, or multiple selections

2.4 MISC1.1 pop complexity **0.5 / 0.5**

✓ + **0.5 pts** **$$O(1)$$**

+ **0 pts** $$O(n)$$

+ **0 pts** No answer, or multiple selections

2.5 MISC1.2 enqueue pointer **0.5 / 0.5**

+ **0 pts** head

✓ + **0.5 pts** **tail**

+ **0 pts** either

+ **0 pts** No answer, or multiple selections

2.6 MISC1.2 enqueue complexity **0.5 / 0.5**

✓ + **0.5 pts** **$$O(1)$$**

+ **0 pts** $$O(n)$$

+ **0 pts** No answer, or multiple selections

2.7 MISC1.2 dequeue pointer **0.5 / 0.5**

✓ + **0.5 pts** **head**

+ **0 pts** tail

+ **0 pts** either

+ **0 pts** No answer, or multiple selections

2.8 MISC1.2 dequeue complexity **0.5 / 0.5**

✓ + **0.5 pts** **$$O(1)$$**

+ **0 pts** $$O(n)$$

+ **0 pts** No answer, or multiple selections

2.9 MISC2 snowAngels **0 / 2**

+ **2 pts** $$1 + 2 \cdot \log_{2}n$$ or equivalent

+ **1.5 pts** $$2 \cdot \log_{2}n$$

+ **1.5 pts** $$log(n^2 + 1)$$

+ **1.25 pts** $$1 + \log_{2}n$$

+ **0.75 pts** $$\log_{2}n$$

+ **0.5 pts** $$O(\log n)$$

✓ + **0 pts** **Incorrect or no answer**

2.10 MISC2 skiHill **0 / 2**

+ **2 pts** $$n-1$$

+ **0.75 pts** $$O(n)$$

+ **0.5 pts** Any other exact or asymptotic answer which is $$O(n \log n)$$

✓ + **0 pts** **Any exact or asymptotic answer which is $$O(n^{2})$$**

+ **0 pts** Incorrect or no answer

2.11 MISC3 post-order reverse of pre-order **2 / 2**

+ **0 pts** always

✓ + **2 pts** **sometimes**

+ **0 pts** never

+ **0 pts** No answer, or multiple selections

2.12 MISC3 reconstruct with pre-order and in-order 2 / 2

✓ + **2 pts** **always**

+ **0 pts** sometimes

+ **0 pts** never

+ **0 pts** No answer, or multiple selections

2.13 MISC3 if complete, post-order and in-order are same 0 / 2

+ **0 pts** always

✓ + **0 pts** **sometimes**

+ **2 pts** never

+ **0 pts** No answer, or multiple selections

QUESTION 3

3-way merge sort 10 pts

3.1 Recurrence expression 2.5 / 3

✓ + **1 pts** **Coefficient of recursive term: $$3$$**

✓ + **1 pts** **Subproblem size: $$M(\frac{n}{3})$$**

+ **1 pts** Non-recursive work: $$+3n$$

✓ + **0.5 pts** **Non-recursive work: +cn or +n**

+ **0 pts** Incorrect or no answer

+ **2 pts** Click here to replace this description.

+ **0.75 pts** Partial marks

+ **0.5 pts** Click here to replace this description.

3.2 M(n) closed form - answer 1 / 2

+ **2 pts** $$n + 3n \log{}_3{n}$$ or equivalent

+ **1 pts** $$3n \log{}_{3} n$$

+ **1 pts** $$3n\log{}_3 n + f(n)$$ where $$f(n)$$ is not of the form $$cn$$

✓ + **1 pts** **$$cn \log{}_3{n} + n$$ where $$c \neq 3$$**

+ **1 pts** $$3n\log{}_3 n + cn$$ where $$c \neq 1$$

+ **1 pts** $$n + n \log{}_{3} n$$

+ **1 pts** $$ n \log{}_{3} n$$

+ **1 pts** $$n \log{}_3 n + \log{}_3 n$$

+ **0.5 pts** $$n + \log{}_3 n$$

+ **0 pts** Expression is recursive, contains a summation, or is asymptotic

+ **0 pts** Other incorrect answer, or no answer

+ **0 pts** $$n$$

+ **0 pts** $$3n$$

+ **0 pts** $$ 3 \log{}_{3} n$$

+ **0 pts** $$ \log{}_{3} n$$

3.3 M(n) closed form - work 1 / 2

+ **2 pts** n + 3nlog_3n

+ **1.5 pts** Correct work with small algebraic/conceptual mistake. Granted if work contains correct generalized substitution (i.e. M(n) = 3^kM(n/3^k) + k3n). Also granted for c*n + d*nlog(n) where c != 1, 0 and d != 3, 0 and work is shown.

✓ + **1 pts** **Shows reasonable substitution (from incorrect or correct answer in part a)**

+ **0 pts** Incorrect, not nlog(n), no work shown or incorrect work

+ **1 pts** Reasonable use of a method other than substitution (i.e. recursion tree) but final answer is incorrect.

+ **0.5 pts** An answer including c*nlog(d*n) + f(n) where c,d are constants and no work shown or incorrect work.

3.4 M(n) tight bound 2 / 2

✓ + **2 pts** **$$\Theta(n \log n)$$**

+ **0 pts** $$\Theta(n)$$

+ **0 pts** $$\Theta(n^2)$$

+ **0 pts** $$\Theta(\log n)$$

+ **0 pts** Other incorrect answer, or no answer

3.5 Asymptotically faster than 2-way? yes/no 1 / 1

✓ + **1 pts** **No**

+ **0 pts** Yes

+ **0 pts** No answer, or other answer

QUESTION 4

Olympic skating 10 pts

4.1 R(4) 1 / 1

✓ + **1 pts** **$$R(4)=14$$**

+ **0 pts** Incorrect or no answer

4.2 R(n) subproblem size 1 / 1

✓ + **1 pts** **$$n-1$$**

+ **0 pts** Incorrect or no answer

4.3 R(n) added partitions 2 / 2

✓ + **2 pts** **$$n-1$$**

+ **1 pts** $$n$$

+ **1 pts** $$\frac{n}{2}$$

+ **0 pts** Incorrect or no answer

4.4 R(n) closed form - answer 2 / 2

✓ + **2 pts** **$$R(n) = n^{2} - n + 2$$ or equivalent**

+ **1 pts** $$R(n) = 2n^{2}-2n+2 - 2\sum_{i=1}^{n-1}i$$ or equivalent
(correct answer but unsimplified summation)

+ **0 pts** Incorrect or no answer

4.5 R(n) closed form - work 4 / 4

✓ + **4 pts** **Correct Answer**

+ **2.5 pts** Shows at least 2 reasonable substitution

+ **1.5 pts** Shows 1 reasonable substitution

+ **0.5 pts** Attempt at generalized form
$$R(n) = R(n-k) ... $$

+ **1.5 pts** Shows a general substituted form:
$$R(n) = R(n-k) + k \cdot 2n - 2\sum_{i=1}^{k}i$$ or similar/equivalent

+ **2 pts** Non-standard, productive supporting work

+ **1 pts** Non-standard, non-productive supportive work

+ **0 pts** Incorrect or no answer

QUESTION 5

Contact lenses 11 pts

5.1 Code trace progress 2.5 / 3

+ **1 pts** Correct shading

✓ + **1 pts** **List doesnt change across iterations.**

✓ + **1 pts** **Correct variables**

+ **0.5 pts** Partial correct variables

✓ + **0.5 pts** **Partial correct shading**

+ **0.5 pts** Partial correct list

5.2 Invariant property (B) 1 / 2

+ **2 pts** $$2(numL-1)$$ or equivalent, e.g. $$2 numL-2$$, $$ 2(k-numR-1)$$

+ **1 pts** $$2numL-1$$

✓ + **1 pts** **$$2 \cdot numL$$**

+ **1 pts** $$numL - 1$$

+ **0.5 pts** $$numL$$

+ **0 pts** Incorrect or no answer

5.3 Invariant property (C) 0 / 2

+ **2 pts** $$2(numR-1) + 1$$ or equivalent, e.g. $$2\cdot numR - 1$$, $$2(k-numL)-1$$

+ **1 pts** $$2 \cdot numR -2$$

+ **1 pts** $$2 \cdot numR$$

+ **1 pts** $$numR - 1$$

+ **0.5 pts** $$numR$$

✓ + **0 pts** **Incorrect or no answer**

5.4 Maintenance, i is even 1 / 1

✓ + **1 pts** **$$2 \cdot numL$$**

+ **0 pts** Incorrect or no answer

5.5 Maintenance, i is odd 1 / 1

✓ + **1 pts** **$$2 \cdot numR + 1$$**

+ **0 pts** Incorrect or no answer

5.6 Termination, numL 0 / 1

+ **1 pts** $$\frac{n}{2}$$
or expressed as contacts.size() / 2

✓ + **0 pts** **Incorrect or no answer**

5.7 Termination, numR 0 / 1

+ **1 pts** $$\frac{n}{2}$$
or expressed as contacts.size() / 2

✓ + **0 pts** **Incorrect or no answer**

QUESTION 6

Sorting properties 11 pts

6.1 Partial sorts (a) 2 / 2

+ **0 pts** A
+ **0 pts** B
✓ + **2 pts** **C**
+ **0 pts** D
+ **0 pts** No answer

6.2 Partial sorts (b) 2 / 2

+ **0 pts** A
+ **0 pts** C
+ **0 pts** B
✓ + **2 pts** **D**
+ **0 pts** No answer

6.3 Partial sorts (c) 2 / 2

+ **0 pts** A
✓ + **2 pts** **B**
+ **0 pts** C
+ **2 pts** D
+ **0 pts** No answer

6.4 Partial sorts (d) 0 / 2

✓ + **0 pts** **A**
+ **0 pts** B
+ **2 pts** C
+ **0 pts** D
+ **0 pts** No answer

6.5 Possible Quicksort pivots 3 / 3

✓ + **0.5 pts** **12**
✓ + **0.5 pts** **21**
✓ + **0.5 pts** **22**
✓ + **0.5 pts** **46**
✓ + **0.5 pts** **73**
✓ + **0.5 pts** **99**
- **0.5 pts** One extra number
- **2.5 pts** Five extra numbers
- **3 pts** Six or more extra numbers
+ **0 pts** Incorrect or no answer
- **1 pts** Two extra numbers
- **1.5 pts** Three extra numbers
- **2 pts** Four extra numbers

QUESTION 7

7 Split in the list 3.5 / 6

+ **6 pts** All nodes linked as specified
- **1.5 pts** Additional pointer declared
- **1.5 pts** list1 or list2 variables accessed
+ **0 pts** Incorrect or no answer
+ **4 pts** Incorrect order of pointer updates
+ **5 pts** Minor error
✓ + **3 pts** **Incorrect or no update of p1 and p2**
- **1 pts** Incorrect dereference
+ **2 pts** No update of "next" fields
+ **3 pts** No update of some next field

+ **0.5** **Point adjustment**

Very close solution. The update of p1 in line 4 does not do what you appear to be intending in the diagrams you've written; in order to change the pointer that goes from node 2->3, you need to set p1->next = p2->next->next. You also need to move p1 to the next node each iteration.

[redacted] Who gets the marks? [1 marks]

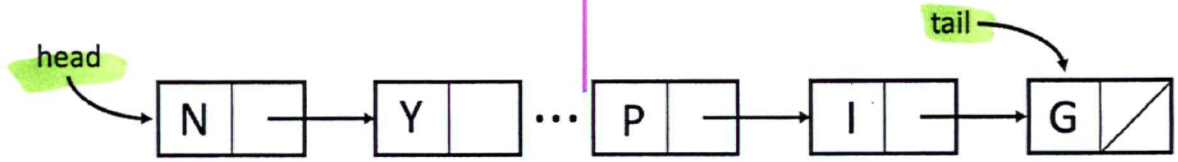
Please enter your 4 or 5 digit CSID in this box: b0o2b

2 A Blizzard [14 marks]

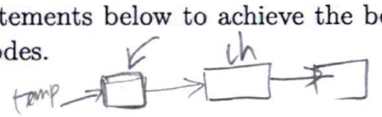
Unless otherwise specified, select the single best answer among the choices.

MISC1 [4 marks]

Suppose we wish to implement a queue and a stack using a singly linked list with head and tail pointers as shown here:



Complete the statements below to achieve the best possible running time for each function, assuming the list contains n nodes.



1. To implement a *stack*, the push function should change (● head / ○ tail / ○ either) pointer, resulting in a running time of (● $O(1)$ / ○ $O(n)$), and the pop function should change (● head / ○ tail / ○ either) pointer, resulting in a running time of (● $O(1)$ / ○ $O(n)$).

2. To implement a *queue*, the enqueue function should change (○ head / ● tail / ○ either) pointer, resulting in a running time of (● $O(1)$ / ○ $O(n)$), and the dequeue function should change (● head / ○ tail / ○ either) pointer, resulting in a running time of (● $O(1)$ / ○ $O(n)$).

MISC2 [4 marks]

In the C++ functions below, give an exact expression for the number of lines printed (times the cou statement is executed), in terms of the input parameter n. Your answer should be a simple function of n. In each case, you may assume that n is a positive power of 2.

```
void snowAngels(int n) {
   for (int i = 1; i <= n*n; i = i*2) {
       cout << "winter is here" << endl;
   }
}
```

n^2

\# lines printed for snowAngels:

$\boxed{\frac{n^2}{2}}$

2 $\frac{n}{2}$

```
int skiHill(int n) {
   for (int i = 1; i < n; i = i*2) {
      for (int j = 1; j <= i; j++) {
         cout << "moguls!!" << endl;
      }
   }
}
```

1+2+3+4+ . . .

$\frac{n(n-1)}{2}$

int n 2 1 1

int n 4 2 3 6

\# lines printed for skiHill:

$\boxed{\frac{n(n-1)}{2}}$

MISC3 [6 marks]

For each of the given statements about binary tree traversals, tell us whether it is always true, sometimes true, or never true, for an arbitrary binary tree containing 2019 nodes.

The post-order traversal is the reverse of the pre-order traversal.

left right root root left right

- ○ always
- ● sometimes
- ○ never

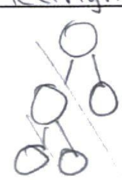
The tree can be reconstructed if you know both its pre-order traversal and its in-order traversal.

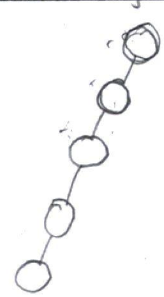
- ● always
- ○ sometimes
- ○ never

If the tree is complete, then the post-order and in-order traversals are the same.

leftright root left root right

- ○ always
- ● sometimes
- ○ never





[redacted] Huey, Dewey, and Louis' Mergesort [10 marks]

The triplets Huey, Dewey, and Louis have invented a new sorting algorithm and want to sell you a copy. The **Mergesort3** algorithm is like **Mergesort** except that it splits the input array into three equal sized parts, rather than just two. It recursively sorts the three parts and then merges the three parts together into one sorted array. **Assume that n is a power of 3.**

Mergesort3($A[1 \ldots n]$)
  if ($n \leq 1$) return A
  $B =$ **Mergesort3**($A[1 \ldots n/3]$)
  $C =$ **Mergesort3**($A[n/3+1 \ldots 2n/3]$)
  $D =$ **Mergesort3**($A[2n/3+1 \ldots n]$)
  return **Merge3**(B,C,D)

merge is linear.

(a) **[3 marks]** Let $M(n)$ be the number of steps taken by **Mergesort3** on an input of size n. Assuming that **Merge3** takes **$3(|B| + |C| + |D|)$ steps** (where $|X|$ means the size of array X), what is the recurrence equation for $M(n)$ expressed using functions of n?

$$M(1) = 1$$

$$M(n) = \boxed{3M(\tfrac{n}{3}) + cn} \quad \text{for } n > 1$$

(b) **[4 marks]** What is $M(n)$ as a function of n? Your solution should not be a recurrence, should not contain a summation, and should not use asymptotic notation.

$$\boxed{n + cn\log_3 n - cn}$$

$$M(n) = 3M(\tfrac{n}{3}) + cn$$
$$= 3\left[3M(\tfrac{n}{9}) + c\tfrac{n}{3}\right] + cn$$
$$2 \quad = 9M(\tfrac{n}{9}) + 2cn$$
$$\vdots$$
$$= 3^k M(\tfrac{n}{3^k}) + kcn$$

base when $\underbrace{\tfrac{n}{3^k}}_{M(1)} = 1$, $n = 3^k$, $\log n = k\log 3 \to k = \log n - \log 3$

$$= nM[1] + [\log_3 n - \overset{1}{\log_3 3}][cn]$$
$$= n + cn\log_3 n - cn$$
$$=$$

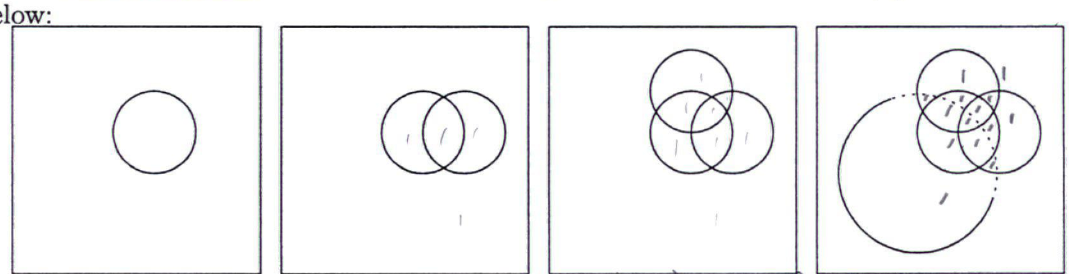
(c) **[2 marks]** Fill in the blank with the simplest function so that the statement is true. (You do not need to prove anything.)

$$M(n) \in \Theta(\boxed{n\log n})$$

(d) **[1 marks]** Is Mergesort3 asymptotically faster than regular old Mergesort (yes or no)? $\boxed{\text{no}}$

4 Olympic skating [10 marks]

The newest Olympic skating preliminary requires skaters to skate perfect circles in the ice. Skaters are rewarded by how many regions they create, where a region is a section of the ice that is bounded by the edges of circles (and the walls of the ice rink). No circle may touch the walls of the rink. Let $R(n)$ be the maximum number of regions that can be created using n circles. The value of $R(n)$ for small values of n is shown below:



$2+2(2-1)$ $\quad$ $4+2(3-1)$ $\quad$ $8+2(4-1)$

$R(1) = 2$ $\quad$ $R(2) = 4$ $\quad$ 4 $\quad$ $R(3) = 8$ $\quad$ 6 $\quad$ $R(4) = \boxed{14}$

$2+2(1)$ $\quad$ $2+2(3)$ $\quad$ $2+2(6)$

(a) **[1 marks]** Fill in the value of $R(4)$ but **be careful!** You should trace the dotted portion of the circle to see exactly how regions are created. If you are surprised by your answer, you're probably on the right track.

(b) **[3 marks]** The arrangement of circles that achieves the maximum number of regions has the property that every pair of circles intersects twice. Given this fact, complete the following recurrence relation for $R(n)$:

$$R(1) = 2$$
$$R(n) = R(\boxed{n-1}) + 2(\boxed{n-1}) \quad \text{for } n > 1$$

(c) **[6 marks]** What is $R(n)$ as a function of n? Your solution should not be a recurrence, contain a summation, or use asymptotic notation. (To check your formula, note that $R(5) = 22$.)

$$\boxed{n^2 - n + 2}$$

$R(n-1) + 2(n-1)$

$= R(n-2) + 2(n-2) + 2(n-1)$

$= R(n-3) + 2(n-3) + 2(n-2) + 2(n-1)$

$\vdots$

$= 2\left(\frac{n(n-1)}{2}\right) + 2$

$= n^2 - n + 2$

Pairing up Contact Lenses [11 marks]

Geoff ordered some contact lenses online. But everything fell apart in shipping, so his left and right contact lenses are all mixed together in the shipping box. He's taken them out and lined them all up into a vector of n lenses (n is even). His left and right eyes have different prescriptions, so he now needs to order the vector of lenses into "L R L R L R ..."

We know that the vector contains $n/2$ each of L and R lenses, so the required output is pretty much known. But Geoff needs to physically swap the lenses. He's written a program to tell him what to do, but he's not sure it works.

```
1  void PairContacts(vector<char> & contacts) {
2    int numL = 0;
3    int numR = 0;
4    for( int k=0; k<contacts.size(); k++ ) {
5      int i = min(2*numL, 2*numR + 1); // index of first unknown contact
6      // Invariant holds here.
7      if (contacts[i] == 'L') {
8        swap(contacts[i], contacts[2*numL]);
9        numL++;
10     } else { // contacts[i] == 'R'
11       swap(contacts[i], contacts[2*numR + 1]);
12       numR++;
13     }
14   }
15 }
```

(a) **[3 marks]** Show the contents of the `contacts` vector at line 6 of each of the iterations of the for-loop in the table below. Indicate the values of `numL`, `numR`, and `i` in the spaces provided. Also lightly shade the vector values that have been processed and are known to be in their correct final locations. The contents at iteration $k = 0$ are given.

	k	0	1	2	3	4	5	numL	numR	i
	0	L	R	R	R	L	L	0	0	0
end of 0	1	L	R	R	R	L	L	1	0	1
end of 1	2	L	R	R	R	L	L	1	1	2
end of 2.	3	L	R	R	R	L	L	1	2	2
				L	R	L	R	1	3	

(b) **[4 marks]** Fill in the blanks for the following loop invariant: At the start of iteration k,

(A) `numL + numR = k`

(B) Even positions in `contacts` upto and including position $\boxed{\text{2numL}}$ contain `L`.

(C) Odd positions in `contacts` upto and including position $\boxed{\text{2numR+1}}$ contain `R`.

(Make sure that the three parts of your invariant are true at the start of iteration $k = 0$.)

(c) **[2 marks]** Suppose that at the start of iteration k the loop invariant is true. We want to show that at the start of iteration $k + 1$ (i.e., the end of iteration k) the loop invariant is true.

Geoff notices that:

If `contacts[i]=='L'` then the swap (line 8) causes `contacts[2*numL]` to become `'L'`.

and

If `contacts[i]=='R'` then the swap (line 11) causes `contacts[2*numR+1]` to become `'R'`.

That's great! But he also notices that `contacts[i]` could become either `'L'` or `'R'` in the swap. We don't know which! That's dangerous!

Convince Geoff that neither swap will ruin your loop invariant by filling in the blanks to argue that i is not one of the positions mentioned in your invariant.

If i is even then $i = \boxed{\text{2numL}}$, which is not in the range of (B).

If i is odd then $i = \boxed{\text{2numR+1}}$, which is not in the range of (C).

(d) **[2 marks]** Let $n =$ `contacts.size()`.

When the for-loop terminates, `numL` equals $\boxed{1}$ and `numR` equals $\boxed{3}$.

(Make sure that your invariant implies that `contacts == ['L','R',...,'L','R']` in this case.)

I sort of know what's going on [11 marks]

The lists below have been partially sorted using some iterative sorting algorithm, completing 3 iterations of the algorithm's outer loop. You must inspect the lists to decide which algorithm(s) could have been used to produce the lists. Assume that the final output is to be in ascending order.

Your available choices are as follows:

(A) This could be produced using Selection sort, but *not* by Insertion sort

(B) This could be produced using Insertion sort, but *not* by Selection sort

(C) This could be produced by either Selection sort or by Insertion sort

(D) This could *not* be produced by either Selection, or by Insertion sort

(a) **[2 marks]** List: 7, 12, 23, 31, 67, 46, 89, 54, 92, 75. Answer: C

(b) **[2 marks]** List: 23, 7, 12, 31, 75, 92, 89, 54, 46, 67. Answer: D

(c) **[2 marks]** List: 23, 46, 75, 54, 67, 12, 92, 31, 7, 89. Answer: B

(d) **[2 marks]** List: 7, 12, 23, 92, 89, 46, 31, 54, 75, 67. Answer: A

(e) **[3 marks]** The partition step of Quicksort picks an array value as the pivot and rearranges the array so that every value smaller than the pivot (and only those values) appear before it. Given the following result of the partition step, write all values that could have been the pivot.

10, 7, 5, 12, 19, 13, 16, 21, 22, 39, 35, 28, 37, 46, 56, 61, 54, 73, 86, 75, 99

Possible pivots: 12, 21, 22, 46, 73, 99

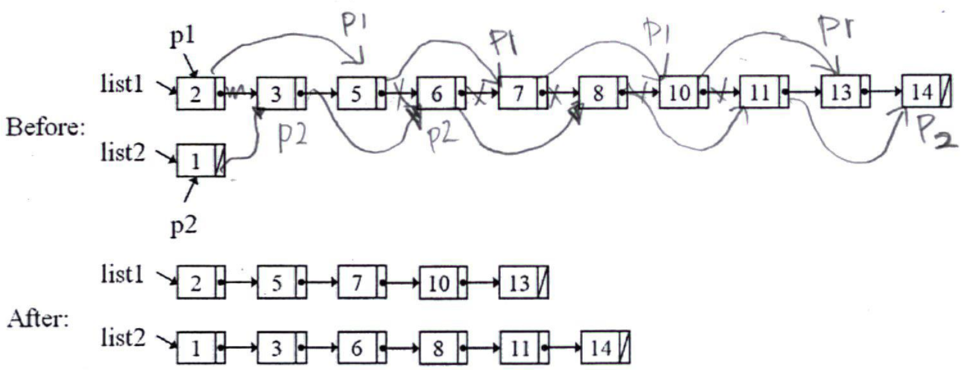
7 A split in the list [6 marks]

In the problem below, we have given you "before and after" models of linked lists. Your task is to transform the "before" into the "after" using simple pointer manipulations on the list nodes. Refer to the elements of the list nodes using the Node class below. Your solution should follow these guidelines:

- Any variables listed in the picture have already been declared and can be used in your solution.
- You may write loops to simplify your solutions, but your answers don't need to be general... they just need to work on the given lists. (Don't worry about even/odd length, or empty lists, for example.)
- Additional restrictions can be found in the individual problem description.

```
class Node {
  public:
    int data;
    Node * next;
    Node(int e): data(e), next(NULL) {} };
```

Perform the necessary transformation, without declaring any additional local pointer variables, without referring to the list1 or list2 variables, and without referring to the data attribute.



line #	
1	
2	while (P2->next != NULL) {
3	p2->next = p1->next;
4	p1 = p2->next->next;
5	p2 = p2->next;
6	}
7	p1->next = NULL;
8	delete p1;
9	delete p2;
10	P1 = P2 = NULL;

// P1 will end at 13.
P2 will end at 14.

This page intentionally left (almost) blank.
If you write answers here, you must CLEARLY indicate on this page what question they belong with AND on the problem's page that you have answers here.

This page intentionally left (almost) blank.

If you write answers here, you must CLEARLY indicate on this page what question th
belong with AND on the problem's page that you have answers here.