Quiz #3 From 20au

STUDENT NAME

Search students by name or email...

Q1 One last time!

0 Points

Welcome to the last quiz of CSE 332 20au! Have fun!

Point values are not final and may be adjusted slightly.

Q1.1 Dance Battle

0 Points

Which staff member do you think would have the best dance moves?

Enter your answer here

Save Answer

Q1.2

0 Points

Prove or disprove P=NP

Enter your answer here

Save Answer

Q2 Parallelism Patterns

15 Points

For problems 2.1-2.5, select the parallelism pattern that would be best suited to tackling each problem.

- reduce
- map operation
- prefix (aka "scan")
- pack (aka "filter")

Select the parallelism pattern ***most appropriate*** for solving the problem and *briefly* **explain why** (a short phrase).

Q2.1 Categorize

3 Points

You are browsing Yelp, looking for a restaurant to order take out from. To make things easier, you would like to categorize each restaurant in a list of restaurants by their cuisine. For simplicity, each restaurant fits into only one cuisine. The output should be a list of pairs, <Restaurant:Cuisine>.

- ○ reduce
- ○ map operation
- ○ prefix (aka "scan")
- ○ pack (aka "filter")

Explanation

Enter your answer here

Save Answer

Q2.2 $$$

3 Points

You've picked a restaurant and now have their menu in front of you. You would like to find the most expensive dish on their menu. If dishes are tied, you can arbitrarily tie break.

- ○ reduce
- ○ map operation
- ○ prefix (aka "scan")
- ○ pack (aka "filter")

Explanation

Enter your answer here

Save Answer

Q2.3 Eat your Veggies!

3 Points

For this meal, you decide to eat vegetarian to get your daily dose of veggies. For convenience, Yelp has flagged all vegetarian dishes with a vegetarian tag. Create a list containing the names of the vegetarian dishes from the menu.

- ○ reduce
- ○ map operation
- ○ prefix (aka "scan")
- ○ pack (aka "filter")

Explanation

Enter your answer here

Save Answer

Q2.4 Just Imagine Waiting...

3 Points

You've ordered your food and start daydreaming about a time when you could go outside and eat at this restaurant. What would that be like? What does a restaurant table feel like anymore? Anyways, you imagine that you are given a list of wait times between each party. How would you find the cumulative wait time for each party?

- ○ reduce
- ○ map operation
- ○ prefix (aka "scan")
- ○ pack (aka "filter")

Explanation

Enter your answer here

Save Answer

Q2.5 Are you gonna eat that dessert?

3 Points

You decide to treat yourself from the dessert menu, but you want to stay under budget. How would you create a list of all the menu items under $10?

- ○ reduce
- ○ map operation
- ○ prefix (aka "scan")
- ○ pack (aka "filter")

Explanation

Enter your answer here

Save Answer

Q3 Parallel Suffix

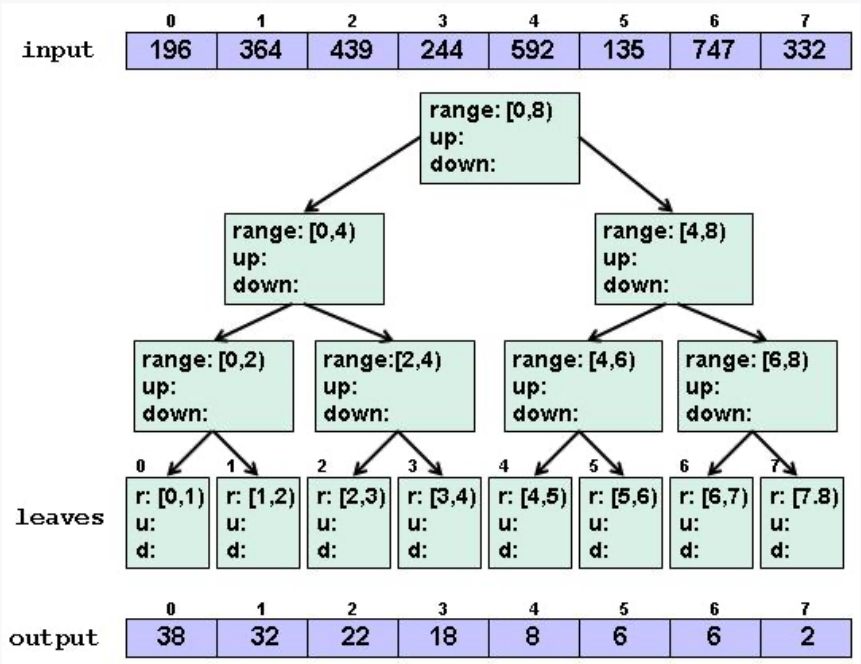
6 Points

For this problem, our `input` is an **array of integers** that are only **3-digit numbers**. We want our `output` array to represent the **sum** of all **even digits** found in the 3-digit numbers from our index to the **right** in the `input` array. In other words, `output[i]` should contain the sum of all even digits in the 3-digit numbers from `input[i]` to `input[input.length - 1]`. Note: that we are **summing** the **digits** that are even, not the values that are even. `input` and `output` are zero-indexed.

This is *similar* to the **parallel prefix** computation we did in class, but from the right instead of from the left. Below is a diagram representing the tree structure. Each node has four fields:

`up` - the field used for the up pass
`down` - the field used for the down pass
`left` - reference to its left child
`right` - reference to its right child
`leaves` - the leaves can be referenced using array indexing, but the leaves are part of the tree, they are not the `output` array.

Fill in each blank with the appropriate calculation. In your calculations, you can refer to things like `input[i]`, `node.down`, `node.left.up`, `leaves[i].down`, etc.



Examples:

```
output[0] = 38
...
output[5] = 4 + 2 + 0 = 6
output[6] = 4 + 2 = 6
output[7] = 2
```

Q3.1 leaves[i].up

1 Point

How would we calculate: (You can describe this algorithm in English if you like)

`leaves[i].up =`

Enter your answer here

Save Answer

Q3.2 node.up

1 Point

How would we calculate: (use pseudocode)

`node.up =`

Enter your answer here

Save Answer

Q3.3 root.down

1 Point

What is the value of: (use pseudocode)

`root.down =`

Enter your answer here

Save Answer

Q3.4 node.left.down

1 Point

How would we calculate: (use pseudocode)

`node.left.down =`

Enter your answer here

Save Answer

Q3.5 node.right.down

1 Point

How would we calculate: (use pseudocode)

`node.right.down =`

Enter your answer here

Save Answer

Q3.6 output[i]

1 Point

How would we calculate: (use pseudocode)

`output[i] =`

Enter your answer here

Save Answer

Q4 Grade Our Assignments Faster

16 Points

The teaching staff is considering a new approach to grading assignments, hoping to leverage concurrency as well as grading assignments as they come in to get grades out as fast as possible (we will not be doing this for this quiz however). For the assignment in question, all 9 TAs will be grading. We have written the following program to help assign student submissions to TAs:

You may want to also look at the explanation for the code, appearing after the code.

```
public class GradingAssigner {
1   private int[] tas = {-1, -1, -1, -1, -1, -1, -1, -1, -1};
2   private int[] submissions = new int[100];
3   private int nextSubmission = 0;
4   private int numUngraded = 0;
5   private ReentrantLock submissionsLock, numUngradedLock, taLock;6
6
7   public void newSubmission(int studentId) {
8      // Add a new submission and ensure each TA has something to grade
9     numUngradedLock.acquire();
10    if (howManyUngraded() > submissions.length) {
11      numUngradedLock.release();
12      throw new UhhSystemIsBrokenException();
13    } else if (howManyUngraded() == submissions.length) {
14      System.out.println("Sorry, please try submitting again later :(");
15    } else {
16      submissionsLock.acquire(); // Add this submission
17      submissions[(nextSubmission + numUngraded) % submissions.length] = studentId;
18      numUngraded = howManyUngraded() + 1;
19      taLock.acquire(); // Make sure each TA has something to grade.
20      for (int i = 0; i < tas.length; i++) {
21        if (tas[i] < 0) {
22          getNextSubmission(taId);
23        }
24      }
25      taLock.release();
26      submissionsLock.release();
27    }
28    numUngradedLock.release();
29  }
30
```

```
31  public void finishGrading(int taId) {
32     // Report that a TA has finished what they were assigned to grade
33     // and assign them something new to grade.
34    taLock.acquire();
35    if (tas[taId] < 0) {
36      System.out.println("This TA is not grading");
37    } else { // Report that TA finished what they were assigned to grade
38      int currStudent = tas[taId];
39      tas[taId] = -1;
40      System.out.println("Submission #" + currStudent + "is ready!");
41      numUngradedLock.acquire()
42      if (howManyUngraded() > 0) { // Assign something new to grade
43        getNextSubmission(taId);
44      }
45      numUngradedLock.release();
46    }
47    taLock.release();
48  }
49
50  public void getNextSubmission(int taId) {
51     // Assign a submission to this TA
52    taLock.acquire();
53    numUngradedLock.acquire();
54    if (tas[taId] >= 0) {
55      System.out.println("TA #" + taId + " is grading submission #" + tas[taId]);
56    } else if (howManyUngraded() <= 0) {
57      System.out.println("All submissions so far have been graded");
58    } else {
59      submissionsLock.acquire();
60      int nextStudent = submissions[nextSubmission];
61      nextSubmission = (nextSubmission + 1) % submissions.length;
62      numUngraded = howManyUngraded() - 1;
63      tas[taId] = nextStudent;
64      submissionsLock.release();
65      System.out.println(taId + " is now grading submission #" + nextStudent);
66    }
67    numUngradedLock.release();
68    taLock.release();
69  }
70
71  public int howManyUngraded() {
72    return numUngraded;
73  }
}
```

Fields

`tas` is an array of size 9, where each TA is assigned an index. The contents at that index will be the student id of the submission the TA should be grading right now or -1 if the TA does not currently have something to grade.

`submissions` holds the list of submissions, where the contents of the array are the student ids corresponding to the submitted assignments.

Methods:

`newSubmission()` will be called whenever a student submits the assignment. This method will add the submission to the end of the queue, and will ensure that each TA is assigned a submission to grade unless there are no submissions left.

`finishGrading()` will be called by the TA after they are done grading their submission. If there are more submissions, the TA will be assigned the next submission in the queue, otherwise, they will be marked as not grading, -1.

`getNextSubmission()` will be called by the TA to get the next submission in the queue, or will let the TA know which submission they are currently assigned to.

`howManyUngraded()` will return the number of ungraded submissions.

For ease of explanation, each method has line numbers.

This series of questions is **not** about debugging the specific methods. It **is** about debugging the concurrency issues. You may assume that each method, if run sequentially, would behave as expected.

Q4.1 Grade Them Quickly

6 Points

Each TA can call `finishGrading()`, `getNextSubmission()`, and `howManyUngraded()` in parallel while the students can call `newSubmission()` in parallel. The teaching staff has started to use locks for our system, as shown above, but we are coming across some concurrency issues. Which of the following concurrency issues are we susceptible to?

- [ ] race condition
- [ ] potential deadlock
- [ ] data race
- [ ] bad interleaving
- [ ] exception thrown
- [ ] none of these

Justify each of your selections; you may need to refer to line numbers in your explanation. For each of the concurrency issues, write at most 3 sentences of justification. **No credit will be given for a selection above if no justification is given.**

Enter your answer here

Save Answer

Q4.2 Grade All of Them

10 Points

Keeping the same locks, restructure where the locks are acquired and released such that the concurrency issues no longer exist as well as making each method as concurrent as possible. You should not modify the methods aside from where the locks are acquired and released.

Feel free to copy and paste the methods from above and change the lines where the locks are acquired and released. (We will ignore line numbers when grading, so if you cut and paste, no need to go back and fix them all.)

Newly modified `newSubmission()`:

Enter your answer here

Newly modified `finishGrading()`:

Enter your answer here

Newly modified `getNextSubmission()`:

Enter your answer here

Newly modified `howManyUngraded()`:

Enter your answer here

Save Answer

Q5 Planet RARSWHJKDH-332

24 Points

332 TAs are secretly astronauts. One day, they land on a new planet, *RARSWHJKDH-332*.

Planet *RARSWHJKDH-332* consists of independent empires. Each empire is made up of multiple cities, connected by bi-directional roads. Inhabitants can get from any city to any other city by way of a route consisting of one or more roads within a empire. However, there is no road connecting two different empires together, since each empire is isolated.

The TAs are given a map of **all** the roads on the planet. The map consists of a list of pairs, in the form: **(cityA, cityB)**. Each pair shows that the city named A is connected to a different city named B by a single road. Each city name is **planetary unique**.

Q5.1 Hello world

6 Points

The TAs are curious to find out how many independent empires there are on the planet. Your task is to design an algorithm to do so, using the planet map given to you.

Describe your algorithm in 3 - 4 sentences.

Enter your answer here

Save Answer

Q5.2 Spaceship landing

6 Points

The TAs choose the empire *Hansa* for further exploration and are given a map containing just the roads for *Hansa*. Interestingly, the roads in *Hansa* do not form a cycle.

The TAs need to find out **which city** in *Hansa* is the best for landing their spaceship.

Upon landing, the TAs will send out robots from the city to explore the rest of the empire. Every single day, a single robot can reach and explore **one** city via one connected road (thus all roads should be considered the same length for this problem). Suppose the TAs have an **infinite** number of robots, the goal is to find the **minimum** number of days for the robots to finish exploring all of the cities in *Hansa*. Design an algorithm to determine which city is the best to land the spaceship.

Describe your algorithm in 3 - 4 sentences.

Enter your answer here

Save Answer

Q5.3 Hit the road

6 Points

The TAs make a tourist stop in the city *Windijef* in the empire *Keshuth*.
Now, they want to start a roadtrip to travel to *Richushi*, a touristy city in *Keshuth*.
Ruchushi is renowned for rich-ness of history and beautiful shi-neries.

For each of the following parts, describe your idea in 1 - 2 sentences.

1. Design an algorithm to find **all possible paths** from *Windijef* to *Richushi*.

Enter your answer here

2. Suppose the TAs can travel to a new city every day, how do you find a path that will take them the least number of days to get to *Richushi* ?

Enter your answer here

3. Now, suppose that the TAs are made aware that the roads are different lengths, and it takes longer time to travel between two cities if the city is further away. How would you change your algorithm accordingly?

Enter your answer here

Save Answer

Q5.4 City exploration

6 Points

The TAs finally arrive at *Richushi*.
They decide to visit some history museums to understand the city's past.
However, they cannot just visit the museums in any order they want, they need to visit them respecting the ordering suggestions of the *Richushi* tourism bureau.

You are given suggestions of the tourism bureau as a list of pairs in the form of **(MuseumA, MuseumB)**.
The pair means that the museum named A needs to be visited before visiting the museum named B.

Here are three pieces of additional information:

1. Each museum name is unique.
2. TAs cannot visit the two museums in the same pair on the same day.
3. Otherwise, they can visit as many museums as they want on the same day.

Your task is to come up with a museum touring plan that minimizes the number of days to visit every single museum in *Richushi*. It is guaranteed that at least one valid schedule exists.

Describe your algorithm in 3 - 4 sentences.

Enter your answer here

Save Answer

Q5.5 Origin story

0 Points

At *Hamvin*, one of the history museums at *Richushi*, the TAs learned about the tale of the planet's origin. Roughly 332 years ago, 10 founding figures summoned a mystery force - *erutcurtsatad* - to burst the planet into existence, and they named the planet after themselves.

Do you have any clue on what the planet name *RARSWHJKDH-332* could possibly stand for?

Enter your answer here

Save Answer

Q6 You're done!

1 Point

Did you work on this quiz alone or collaborate with others? If you collaborated with others, who did you collaborate with? **Please list full names and UWNetIDs of everyone you collaborated with.**

Enter your answer here

How long did you personally spend on this quiz?

Enter your answer here

CONGRATULATIONS! You have now completed CSE 332 20au! Have a wonderful Winter break! Stay Safe!

The CSE 332 Staff

Save Answer