DSA in-class programming activity

2024.4.23

Goal

Objectives when taking the DSA course:

- How to implement DS
- How to identify which DS to use
- How to understand application requirements and design appropriate DS
- How to work with colleagues via a good definition of APIs

Logistics

- Date: 2024.4.23
- Time: 13:20~16:20
- Location: You decide!
- Preparation (for every participant):
  - A laptop
  - A stable Internet connection
  - Technical knowledge learned from DSA (possible)

Roles

1. Project manager
2. Research & Development
    a. Algorithm engineer
    b. Software engineer
3. Quality assurance

Roles - PM

- Project Manager
- Tasks
  - Communicate with the customers (TAs) to find out what to implement
  - Communicate with the software engineers to convey the requirements
  - Get a handle on the progress of your team
    - Write a report about the progress of each of your team members every 30 minutes
  - Write a user manual

Roles - RD

- Research & Development
- Tasks
  - Implement the system! (i.e., write the code)
  - Different orientations
    - Algorithm engineer
    - Software engineer

Roles - RD - Algorithm engineer

- Tasks
  - Implement the DS & A
  - Create API for better “parallel programming”
  - Survey the different implementations of the data structure and algorithms
  - Remember to write a README so that everyone can understand how to use your code!
    (Not a part of grading but we need to know how to run your code in order to grade your code)

Roles - RD - Software engineer

- Tasks
  - Utilize the API to create software that can actually be used by users
  - Design different types of test data and write a test bench to test if the following aspects adhere to the demands of the customer.
    - How to receive inputs from users
    - What to output
    - Make sure the program won’t crash even if some invalid input is processed
    - …
  - Remember to write a README so that everyone can understand how to use your code!
    (Not a part of grading but we need to know how to run your code in order to grade your code)

Teams

- We will team you up
- The team list is already announced on NTU COOL
- You can/should contact your teammates and discuss about how to set the roles.
- Important: If you decide to withdraw from the course, please let us know ASAP

Teams

- 6 people：1 PM, 2 AE, 1 SE, 2 QA
- 5 people：1 PM, 1 AE, 1 SE, 2 QA
  (if one of your members does not participate)
- 4 people：1 PM, 1 AE, 1 SE, 1 QA
  (if two of your members does not participate)
- Less than 3 people：we will merge your teams with others.

Score

- Full credit: 100 points
- If you participate in this activity, **you will receive <u>80 points</u>!**
- The r**emaining 20 points** will be:
  - 12 points of team score
  - 8 points of personal score

Score - personal score

- PM
  - User Manual 40%
  - Requirement communication 30%
  - Progress reports 30%
- AE.1 (part 1 of the algorithm)
  - The simplest implementation 50%
    (To be announced with the actual project)
  - Survey report 25%
  - Any other implementation 25%
- AE.2 (part 2 of the algorithm)
  - The simplest implementation 50%
    (To be announced with the actual project)
  - Survey report 25%
  - Any other implementation 25%
- SE
  - Program that adheres to the spec. 75%
  - Invalid input handling 25%
- QA
  - Performance evaluation report 30%
    (Including the given executable and those written by RDs)
  - Robustness of the test data 30%
  - Correctness of the test data 30%
  - Test bench script 10%

For more detailed rubrics, please refer to your corresponding guidelines
(Guidelines not yet announced)

Score - team score

- Team ranking score = avg. of all team members' personal scores
- We'll rank all teams using team ranking score
- Team ranking of:
  - 0~25% receives 12 points
  - 26~50% receives 8 points
  - 51%~75% receives 4 points
  - 76%~100% receives 0 points

  as your team score (ONLY 20% of the total)

Appendix - API

- Application Programming Interface
- A unified interface that can be used throughout the whole project
- Example: queue.h

Appendix - A

- Application
- A unified int
- Example

ect

```c
#ifndef QUEUE_H
#define QUEUE_H

typedef struct Queue Queue;
struct Queue {
    /* define any field you need here */
};

void queue_init(Queue *queue);
void queue_destroy(Queue *queue);

bool queue_enqueue(Queue *queue, int value);
int queue_dequeue(Queue *queue);

bool queue_empty(const Queue *queue);
bool queue_full(const Queue *queue);
int queue_size(const Queue *queue);

#endif
```

Appendix - Example project

- In this problem, we ask you to design a system to emulate the job queues of several printers. (Yes, this is from HW2)
- Provided materials:
  - queue.h
  - An executable written by TA (which implements the queue.h headers)

Appendix - Example project

- PM’s job
  - Ask what operations are needed, possibly will get the following responses:
    - Add Operation: Insert a job into the printer’s queue.
    - Print Operation: Print the document with the highest priority from the queue.
    - Move Operation: Transfer all jobs from one printer to another.
  - Ask the input/output format.
  - .....
  - Convey the questions and requirements from/to SE
  - Also the reports and manuals.

Appendix - Example project

- AE’s job
  - Implement the functions defined in queue.h in at least one way
  - Do a survey on how queue could be implemented in different ways and write a report about them
  - If you still have time, implement the queue in a better way

Appendix - Example project

- SE’s job
  - Communicate with your PM and build a complete software for the customer
  - Assume the API works and build the application based on it
  - Remember to handle all possible cases that a user might encounter (Always assume users are evil)

Appendix - Example project

- QA’s job
  - Analyze the time/space complexity of the given executable (written by TA) by experimenting using different number of inputs and write a report about the result
  - Design test cases to cover all possible scenarios and test if the program written by RDs can handle them well
  - Evaluate the performance of the program by feeding different number of inputs and measure the computation time/memory consumption.