Q1

(1)

**module** *GitHubRepository*

**exports**

**var** *PullsCode*: string;

**var** *TextReport*: file **of** char;

The ADT consists of a pulls code and a text report.

**end** *GitHubRepository*

(2)

**module** *ProjectManagement*

**uses** *EmailNotification*, *GitHubRepository*

**exports**

**var** *TeamID*: integer

**type** *StudentNames*: **array** (1..5) **of** string;

**type** *StudentIDs*: **array** (1..5) **of** integer;

**procedure** *TeamRegistration*(*StudentNames* **in** **array** (1..5) **of** string; *StudentIDs* **in** **array** (1..5) **of** integer)

**procedure** *SubmitGitHubURL*(*TeamID* **in** integer, *GitHubURL* **in** string)

**procedure** *SubmitDocument*(*TeamID* **in** integer, *DocumentType* **in** integer, *File* **in** file **of** char)

**procedure** *PullCode*(*TeamID* **in** integer, *PhaseType* in integer)

**procedure** *ReleaseMark*(*TeamID* **in** integer, *PhaseType* **in** integer, *Mark* **in** integer, *Comment* **in** string)

implementation

*TeamRegistration* first check whether there is conflict on student names with another registered team. If it returns no conflict, a team ID is generated, and it is sent to all team members by *EmailNotification*.

*SubmitGitHubURL* once received team ID and the GitHub URL, it sends email notification to all team members by *EmailNotification*.

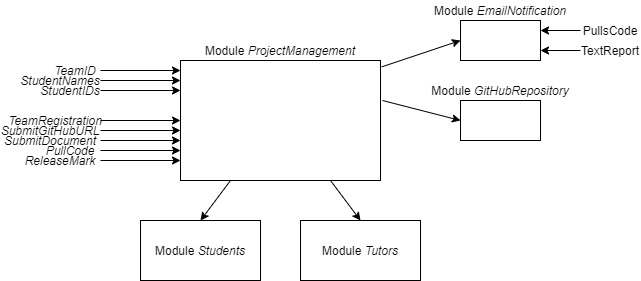
*SubmitDocument* once received team ID and the document, it sends email notification to all team members by *EmailNotification*.

*PullCode* once received team ID and phase type, it will pull the code of the team, generate a status report, and send email to all team members by *EmailNotification*.

*ReleaseMark* once received team ID, phase type, marks, and comments, it will send email to all team members by *EmailNotification*.

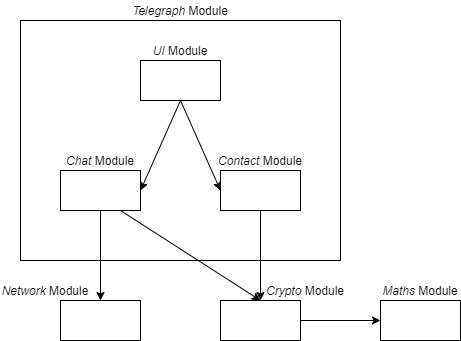
**end** *ProjectManagement*

(3)

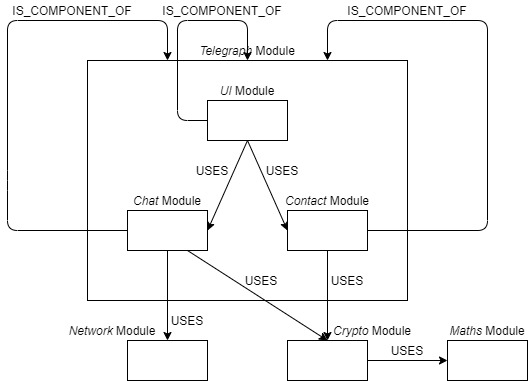


Q2

(1)



(2)



(3)

, , so is not defined.

, , so

, , so

, , so

, , so

, , so

, , so

(4)

“Stability” measures the maintainability of a module.

If a module has high stability, that module should be unlikely to change because it has many incoming dependencies. It is not easy for us to modify the module, so the maintainability is low. Therefore, we can use stability to measure the quality of the module in maintainability aspect.

Q3

(1)

Adjacency list

|  |  |
| --- | --- |
|  | L[] |
| 1 | [5, 6] |
| 2 | [3] |
| 3 | [2, 4, 6] |
| 4 | [2] |
| 5 | [1, 2, 4] |
| 6 | [1, 4, 5] |

Adjacency Matrix

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| \ | 1 | 2 | 3 | 4 | 5 | 6 |
| 1 | 0 | 0 | 0 | 0 | 1 | 1 |
| 2 | 0 | 0 | 1 | 0 | 0 | 0 |
| 3 | 0 | 1 | 0 | 1 | 0 | 1 |
| 4 | 0 | 1 | 0 | 0 | 0 | 0 |
| 5 | 1 | 1 | 0 | 1 | 0 | 0 |
| 6 | 1 | 0 | 0 | 1 | 1 | 0 |

(2)

Step 1

Assuming there are 𝑉 vertices.

**function** CheckAcyclic()

i := 1;

**while** i <= 𝑉 **loop**

If the graph is null, it is acyclic. If there exist some modules which do not USES another modules, we remove such module, then continue to loop.

i := i + 1.

**end** loop;

**return** false; // If all modules USES another module, the graph must be cyclic, return false.

**end** **function**

Step 2

Assuming there are 𝑉 vertices.

**function** CheckAcyclic()

i := 1; module:= V; remove = null;

**while** i <= 𝑉 **loop**

**if** module == 0

**return** true;

**end if**

**if** i **belongs to** remove

i := i + 1;

**continue**

**end if**

**if** length(L[i]) == 0

remove = [remove i];

module = module – 1;

i = 1;

**continue**

**end if**

i = i + 1;

**end** loop;

**return** false;

**end** **function**

We will loop the i while loop for V times. In each while loop, if we remove a module, we run the loop again, then we loop for V – 1 times.

Totally, we need V + (V – 1) + … + 1 = V\*(V+1)/2, which is a O() process.

Q4

(1)

一張含有 文字 的圖片

自動產生的描述

一張含有 文字 的圖片

自動產生的描述

(2)

1. pDistance

2. points[idx]

3. points[0]

4. dist

5. maxDist

(3)

1. 1

2. maxDist > threshold

3. points[1…maxIdx]

4. points[maxIdx…-1]

5. false

(4)

1. -1

2. stack.StackIsEmpty()

3. maxDist > threshold

4. e[0]

5. e[1]

6. e[0]

7. maxIdx