Software Design & Development 2018 Trial Paper Solutions

Compiled (haha) by JTrudeau and Opengangs

Section I

Q1	A
Q2	A
Q3	B
Q4	A
Q5	D
Q6	C
Q7	B
Q8	B
Q9	D
Q10	D
Q11	B
Q12	C
Q13	D
Q14	A
Q15	A
Q16	C
Q17	B
Q18	C
Q19	A
Q20	D

Section II

Question 21

Reverse Engineering is the process of analysing a software's parts with the intention of recreating it, while Decompilation is translating the machine code into a higher level language (usually assembler) with the intention of acquiring the original source code. RE is done by a programmer, while decompilation is done by a decompiler/computer.

Question 22

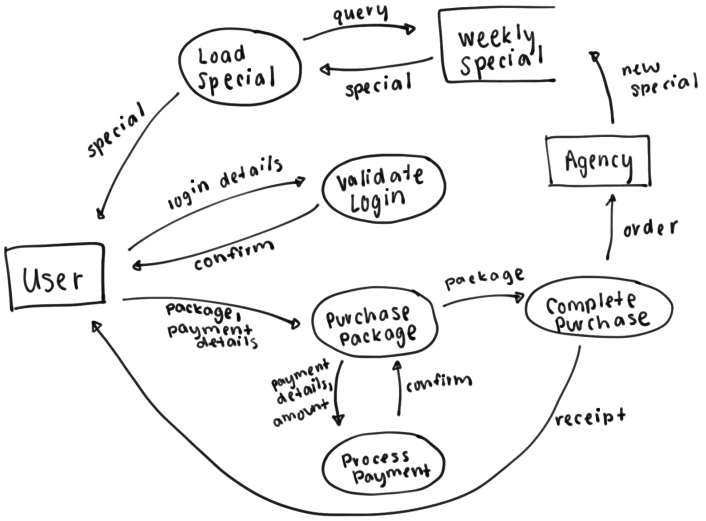
1. Security measures-- back to base authentication (validating user's software id every time the software is run to ensure up-to-date account), non-copyable data sheets, registration codes, etc.
2. Government-- fines of $500 per item pirated, $2500 for corporations
3. Education-- users need to be educated on copyright laws, starting with the youth.

Question 23

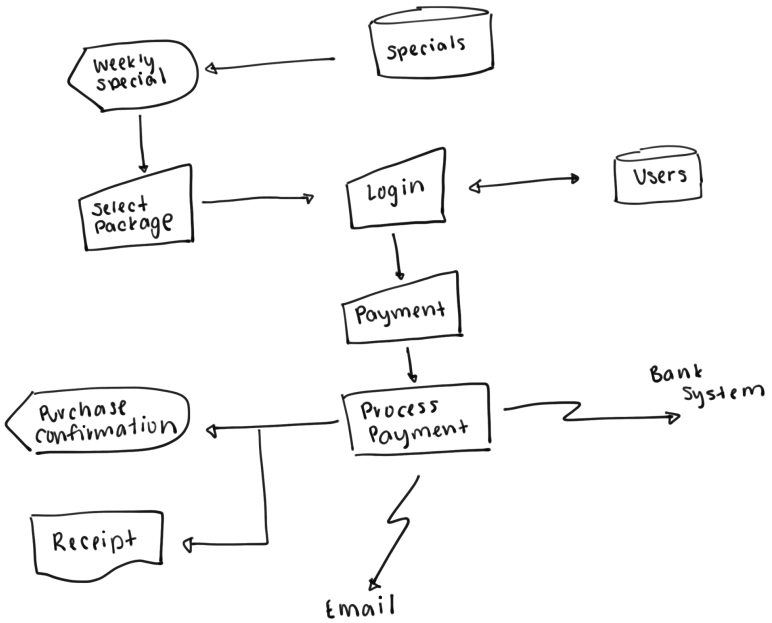
Peter is correct. Tom used bubble sort instead.

Question 24

a)



b)



Question 25

a)

- Processor speed
- RAM size/type
- Secondary storage size/access speed
- Input and output devices
- Network architecture
- Drivers and extensions

b)

A communication plan details the frequency and method used by the team to exchange information, provide feedback to users, and obtain feedback from users. Methods of communication may include emails, phone calls, diagrams, and algorithms.

The defining and understanding stage of the software development cycle involves significant consultation with the client, and sets the foundation for the project's success. Without properly understanding the client's needs and specifications, the resulting program will not efficiently meet the client needs and so may lead to subpar software or project failure.

By involving the client in the development process, the client/end user can pick up problems/workarounds that are missed by the developers because they are too 'close' to the project. The implementation of the project will also be smoother if the developers frequently communicated changes to the end users, who are fully informed and have been involved in the development.

Note: Answers may also include a discussion of communication between software developers in an agile environment.

Question 26

a)

Neither Molly nor Emma are allowed to sell their products on the app store. While shareware means parts of the software are free to use, decompilation, derivatives, and modifications are not allowed and so Molly cannot sell her product.

Emma's freeware is also free to use, and decompilation, derivatives, and modifications are allowed, but she can only distribute the derivative product as freeware. She too, is not allowed to sell her product.

b)

A developer would use interpretation to test specific lines of source code for syntax and runtime errors, usually in an IDE. A developer uses compilation to produce an executable that runs faster and more efficiently, and can be run repeatedly without need for recompilation, thus guaranteeing a consistent quality for the end user.

c)

Malware (malicious software) are programs designed to damage or perform unwanted actions on computer systems. They can disrupt computer operation, gather sensitive information, and gain access to private systems, causing damage to hardware or data and sometimes financial loss. Examples of malware include trojans, worms, and spyware.

Question 27

a)

Identifier Name	Data type	Size	Description
StoreName	String	50 characters	The name of the bagel store
Address	String	200 characters	The unit number, house number, street name, and postcode of the store
Rating	Int	3 bits	A number between 0 and 5
Average Price	Float	8 bytes	The average price of a bagel in dollars (2 decimal places)
InManhattan	Boolean	1 bit	True if the store is in Manhattan, false if it isn't

b)

Jacob could use either End User Development or Rapid Applications Development (RAD). End User would fit his low budget, as he doesn't need to hire a professional developer. Instead, he can build the app the way he wants (since he knows his own user needs). However, the resulting app may be inefficient or lower quality, as he isn't a professional developer.

RAD is often faster and can be done on a low budget. It has a standardised look and feel due to APIs and other reusable components, and CASE tools may reduce the errors in the code, but there may be legal problems due to third party components and the resulting product is less efficient as it isn't a custom-built solution.

Since Jacob is planning to release the app out to the public for other people to use, he should have a professional developer work on it, and so RAD is the recommended option.

c)

```
BEGIN FixIt(BagelGuide)
        i = 1
        FOR i TO BagelGuide.length STEP 1
                BagelGuide[i].StoreName = DeleteScott(BagelGuide[i].StoreName)
                BagelGuide[i].Address = Trim(BagelGuide[i].Address)
        ENDFOR
END FixIt

BEGIN DeleteScott(oldString)
        newString is an array of characters of length (oldString.length -5)
        i = 1
        FOR i TO 3 STEP 1
                newString[i] = oldString[i]
        ENDFOR

        i = 9
        FOR i TO oldString.length STEP 1
                newString[i-5] = oldString[i]
        ENDFOR

        RETURN newString
END DeleteScott

BEGIN TrimString (oldString)
        start = 0
        end = oldString.length + 1
        i = 1
        exitFlag = False
        WHILE NOT exitFlag DO
                IF oldstring[i] == " " THEN
                        start += 1
                ELSE
                        start += 1
                        exitFlag = True
                ENDIF
                INCREMENT i
        ENDWHILE

        exitFlag = False
        j = oldString.length
```

```
        WHILE NOT exitFlag DO
            IF oldString[j] == " " THEN
                end -= 1
            ELSE
                end -=1
                exitFlag = True
            ENDIF
            DECREMENT j
        ENDWHILE

        newString is an array of characters of length (end - start + 1)
        x = 1
        FOR k = start TO end
            newString[x] = oldString[k]
            INCREMENT x
        ENDFOR

        RETURN newString
End TrimString
```

Question 28

a)

There are three steps: Lexical Analysis, Syntactical Analysis, and Code Translation. In Lexical Analysis, the computer examines each word to see if it's a valid part of the language. Lexemes are checked against the symbol table and assigned a token if it matches an element. If no match is found, an error is raised.

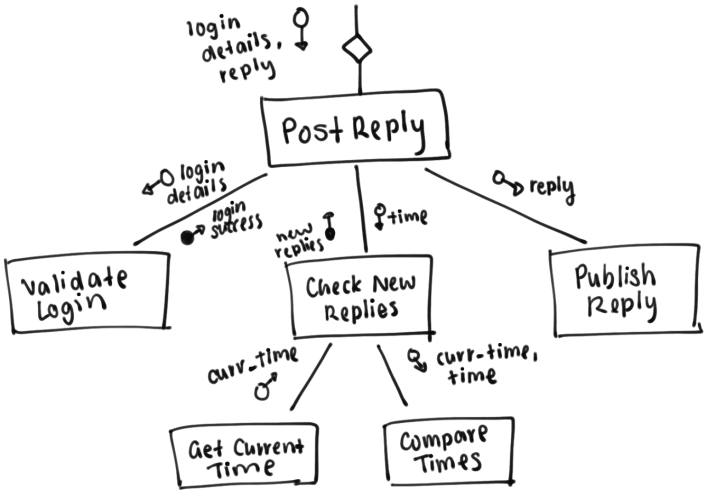
In Syntactical Analysis, the computer checks the syntax of a code statement by arranging the tokens into a parse tree, which then undergo type check. If a series of tokens cannot be parsed, an error is raised.

In Code Translation the computer traverses the parse tree and converts tokens to machine code. If a compiler is used, then the machine code is stored in an object file. A Linker links the object file to runtime libraries and the DLL to produce the final executable.

b)

```
BEGIN PrintSixUniqueLottoNumbers
        FOR i = 1 TO 45
                Flag[i] = 0
        NEXT i
        FOR i = 1 to 6
                REPEAT
                        r = RAND(1, 45)
                UNTIL Flag[r] = 0
                Flag[r] = 1
                Display "Your next number is" r
        NEXT i
END PrintSixUniqueLottoNumbers
```

Question 29


login details, reply
Post Reply
login details
login success
new replies
time
reply
Validate Login
Check New Replies
Publish Reply
curr_time
curr_time, time
Get Current Time
Compare Times

Section III

Question 30: Programming Paradigms

a) **Abstraction** refers to the **hiding of unnecessary details from the user**, while **encapsulation** refers to the **hiding of properties inside a class**. When data is encapsulated, it may only be accessed by appropriate methods and can generally never be accessed by unauthorised users (ie. **private data**). However, with abstraction, users will have the functionality information for particular methods so they are free to change relevant information (ie. **public data**). Thus, abstraction is developed during the **design stage**. Encapsulation is developed during the **implementation stage**.

b)

i) The **barcode_id** has been implemented publicly. This means the end user may change it as they wish. In doing so, data that should never generally be changed may be accidentally changed - thus, the piano and guitar barcode may have been changed to the triangle barcode, which implies that the piano and guitar have been charged the same as the triangle. To fix this, the barcode_id should be private data as this should never be accessed by the end user to preserve the price. Also, the price should be private data as this is generally never touched by the end user.

ii)
```
instrument = new MusicInventory();
instrument.instrument_type = “Sitar”;
instrument.working = true;
instrument.numbers_supplied = 350;
instrument.supplier = “BTAR Notes Ltd.”;
instrument.price = 350;
instrument.barcode_id = 540948;
```

c)

i)
```
backwards(X, Y) :- straight(X, Y), left(X, Y), backwards(X, nil);
right(X, Y) :- straight(X, Y), left(X, Y), backwards(X, Y), right(X, nil);
```

ii) fight_back(X, Y, Z) :- robot(Y), fight(X, Y, 4), (straight(X, Y); left(X, Y); right(X, Y); backwards(X, Y)); - here, we claim that Y is a robot. fight(X, Y, 4) implies that, because Y is a robot, then if a robot is within a 4

circle radius of the AI, then fight will be successful. The hardest part is to face the right way. We need to detect if the robot is either:

- In front of us (straight)
- To our left (left)
- Behind us (backwards)
- To the right of us (right)

Knowing this, we can then detect which way to face and it's important to enclose this all together (as they are being called **together**, not separately).

d) **Logic paradigms** are a paradigm largely based on logic and inductive reasoning. Logic paradigms often use a combination of rules (set by the program through forward/backward chaining) and facts (set by the developer). As such, these are primarily based around artificial intelligence where logic and reasoning are the primary fundamentals.

On the other hand, **object oriented paradigms** are largely based around repetition and actions, rather than the logical steps. Objects largely contain data and attributes, and thus, may need to have copy-pasted data (instantiation). Object oriented paradigms are used more commonly inside programming languages, such as C, C++, Java, and are geared more towards application-based software where information and actions are continuously updated.

Question 31: The Interrelationship Between Hardware and Software

a)

i) Both One's Complement and Two's Complement are used to represent negative numbers in binary and overcome the shortfalls of the Sign/Modulus method. One's Complement takes the binary number in positive form and reverses all digits. E.g. 23 is 00010111, so -23 is 11101000

Two's Complement is more efficient for the computer, and has the same method as One's Complement, and then adds 1 to the rightmost bit. So -23 becomes 10010111.

ii) Negative → 1 for the sign bit
51.625 = 32 + 16 + 2 + 1 + 0.5 + 0.125
= 00110011.101 in binary
0.00110011.101 → 1.10011101×2^5
Mantissa = 10011101 and trailing 0's
Exponent 5 + bias = $101 + 2^{8-1} - 1$
= 101 + 01111111
= 10000100

Final answer:
1 10000100 10011101000000000000000

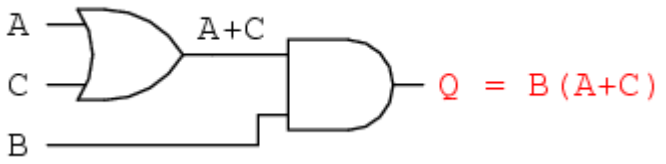
b)

i) $(AB) + ((B + C)(BC))$
$(AB) + (BBC + BCC)$
$AB + BC + BC$
$AB + BC$
$B(A + C)$

ii)

A	B	C	AB	B+C	BC	(B+C)(BC)	Q
0	0	0	0	0	0	0	0
0	0	1	0	1	0	0	0
0	1	0	0	1	0	0	0
0	1	1	0	1	1	1	1
1	0	0	0	0	0	0	0
1	0	1	0	1	0	0	0
1	1	0	1	1	0	0	1
1	1	1	1	1	1	1	1

iii)



c) A flip flop is a bistable device used to store binary digits. It has 3 basic components: Latches to store data, a Clock to ensure changes occur at the same time, and an Edge to ensure the flip flop only changes state once every tick. It works using a feedback loop with two NAND or NOR gates

d)

i) A = 65, a = 97 in ASCII
4A = 0100 1010 = J
6F = 0110 1111 = o
57 = 0101 0111 = W
61 = 0110 0001 = a
Plan JoWa

ii) The data block sent by a text messaging app would only contain text and control characters for formatting, usually up to a certain character limit. Due to this, the header will be comparatively small, as there are fewer instructions on how to interpret the data block. The trailer will contain error checking parity bits to ensure the message is

transmitted correctly.

The CCTV, on the other hand, is transmitting video feeds, which are much larger and sent more frequently (to give the illusion of motion). Because the images are sent more frequently, fewer error checking bits are required, but they will be needed to ensure the images are being sent correctly. The header will have more information contained to indicate how to interpret the data block, and the size of the data block will be larger.