

Must have рівень:

1. Зроби порівняння статичних та динамічних технік тестування. Наведи переваги та можливі обмеження.

	Статистична техніка тестування	Динамічна техніка тестування
Основна інформація		
Перевага №1	Статичне тестування проводиться без запуску програми (тобто до компіляції коду)	Динамічне тестування усуває вже знайдені дефекти
Перевага №2	Дозволяє проаналізувати вимоги на початку	Динамічне тестування включає тестові кейси для виконання
Перевага №3 (і т.д.)	Статичне тестування запобігає появі дефектів	Динамічне тестування може бути автоматизовано за допомогою спеціальних інструментів.
Обмеження №1	Потребує спеціальних знань коду	Дозволяє нам знайти дефекти в системі, коли код вже написаний
Обмеження №2	Процес статичного тестування може займати багато часу, так як в основному він виконується вручну.	Вартість пошуку та виправлення дефектів висока
Обмеження №3 (і т.д.)	Перешкоджає виявленню вразливостей, представлених в середовищі виконання.	Висока вартість проведення тестування
Висновок	Щоб знайти помилки раніше, і щоб це було дешевше підійде статичне тестування	Динамічне важливо провести теж перед тим, як ми віддамо продукт на прод.

Середній рівень:

1. Виконай завдання попереднього рівня.
2. Наступне твердження стосується покриття рішень:  
*Коли код має одну 'IF' умову, не має циклів (LOOP) або перемикачів (CASE), будь-який тест, який ми виконаємо, дасть результат 50% покриття рішень (decision coverage).*

Яке твердження є коректним?

- a. Коректно. Будь-який тест кейс надає 100% покриття тверджень, таким чином покриває 50% рішень.
- b. Коректно. Результат будь-якого тесту умови IF буде або правдивим, або ні.
- c. Некоректно. Один тест може гарантувати 25% перевірки рішень в цьому випадку.

d. Некоректно, бо занадто загальне твердження. Ми не можемо знати, чи є воно коректним, бо це залежить від тестованого ПЗ.

3. Є псевдокод: Switch PC on -> Start MS Word -> IF MS Word starts THEN -> Write a poem -> Close MS Word.

Скільки тест кейсів знадобиться, щоб перевірити його функціонал?

- a. 1 – для покриття операторів, 2 – для покриття рішень
- b. 1 – для покриття операторів, 1 – для покриття рішень
- c. 2 – для покриття операторів, 2 – для покриття рішень
- d. 2 – для покриття операторів, 1 – для покриття рішень

4. Скільки потрібно тестів для перевірки тверджень

Read P

Read Q

IF  $P+Q > 100$  THEN

Print "Large"

ENDIF

If  $P > 50$  THEN

Print "P Large"

ENDIF

коду:

- a. 2
- b. 1
- c. 3
- d. 4

Програма максимум:

1. Виконай завдання двох попередніх рівнів.
2. Продовжуємо розвивати стартап для застосунку, який дозволяє обмінюватися фотографіями котиків.

Є алгоритм:

*Запитай, якого улюбленця має користувач.*

*Якщо користувач відповість, що має kota, то запитай, яка порода його улюбленця: «короткошерста чи довгошерста?»*

*Якщо клієнт відповість «довгошерста», то запитай: «ви бажаєте отримати контакти найближчого грумера?»*

*Якщо клієнт відповість «так», то скажи: «Надайте адресу найближчої котячої перукарні»*

*Інакше*

*Скажи: «Запропонуй магазин з товарами по догляду за шерстю»*

*Закінчити*

*Інакше*

*Скажи «Запропонуй обрати магазин із зоотоварами»*

*Закінчити*

*Якщо клієнт не має kota*

*Скажи “Коли вирішите завести улюбленця – приходьте”*

*Закінчити*

Завдання:

1. Намалюй схему алгоритму (в інструменті на вибір, наприклад, у вбудованому Google Docs редакторі, [figjam](#) чи [miro](#))
2. Який потрібен мінімальний набір тест-кейсів, щоб переконатися, що всі запитання були поставлені, всі комбінації були пройдені та всі відповіді були отримані?

1. Якого ми маєте домашнього улюбленця?

Клієнт: маю kota

