Must have рівень:

1. Зроби порівняння статичних та динамічних технік тестування. Наведи переваги та можливі обмеження при використанні кожної з них.

	Статична техніка тестування	Динамічна техніка тестування
Основна інформація	Є методикою тестування програмного забезпечення, при якій ПЗ тестується без запуску коду.	Тип тестування, який перевіряє функціональність програми, коли код виконується.
Перевага №1	Знижує вартість фіксу знайдених багів, оскільки виявляє баги на ранніх етапах циклу розробки програмного забезпечення	Це ретельне дослідження, яке розглядає всю функціональність програми, тому якість відповідає найвищим стандартам
Перевага №2	Відгуки, отримані в ході цього тестування, допомагають покращити функціонування процесу, що також допомагає команді уникнути подібних дефектів і багів	Процес динамічного тестування добре налагоджений, додаток тестується з точки зору користувача, що підвищує якість ПЗ
Перевага №3	Підвищує інформованість про різні проблеми якості програмного забезпечення	Виявлення складних помилок, які могли вислизнути на етапі код рев'ю
Перевага №4	Покращує обмін критичної і важливої інформації між членами команди	Динамічне тестування може бути автоматизовано за допомогою спеціальних інструментів

Перевага №5	Істотно скорочуються зусилля по виправленню помилок, що ще більше сприяє продуктивності розробки	
Обмеження №1	Витрати часу. При правильному виконанні статичне тестування може заощадити командам багато часу. Однак, це вимагає витрат часу, які можуть бути особливо обтяжливими, коли це робиться вручну для складних збірок програмного забезпечення	Оскільки динамічне тестування являє собою складний процес, воно займає багато часу
Обмеження №2	Обмежена сфера застосування. Перешкоджає виявленню вразливостей, представлених в середовищі виконання	Висока вартість проведення тестування
Обмеження №3	Залежність від людського втручання. Ручне статичне тестування значною мірою залежить від навичок і досвіду тестувальників. Якщо людина-рецензент не має достатніх навичок, досвіду та знань, вона може легко пропустити дефекти та помилки, що нівелює деякі переваги статичного тестування.	Динамічне тестування зазвичай виконується після завершення кодування, і знайдені баги виявляються пізніше в життєвому циклі розробки

Висновок	Незважаючи на те, що **статичне тестування** вимагає багато часу на бурхливі дискусії та зустрічі, все ж варто витратити час на запобігання появи дефектів на останніх етапах розробки продукту. Тому **статичне тестування** по праву вважається важливим кроком на шляху до розробки ПЗ без помилок. Але важливість **динамічного тестування** також величезна. Завдяки безпосередньому виконанню тестів програмного забезпечення (перевірки функціональної поведінки, продуктивності, надійності та інших важливих аспектів) команда може перевірити і підтвердити якість і ефективність ПЗ

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 правдими, 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 коду:

```
Read P
Read Q
IF P+Q > 100 THEN
Print “Large”
ENDIF
If P > 50 THEN
Print “P Large”
ENDIF
```

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 кота, 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 кота
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?