**Перший рівень**

**Завдання 1**

|  | **Статична техніка тестування** | **Динамічна техніка тестування** |
| --- | --- | --- |
| **Основні характеристики** | Етап верифікації ПЗ  Не потребує виконання програмного коду  Орієнтоване на запобігання дефектів  Виконується на ранніх етапах розробки програмного забезпечення  Менша вартість виправлення багів  Забезпечує більш ширше охоплення,аніж динамічне, за коротший проміжок часу.  Включає в себе різні методи оцінки, наскрізний перегляд, рев’ю і багато іншого.  Мета – запобігання дефектів програмного забезпечення  Комплексне тестування коду, яке допомагає знайти більше дефектів у системі.  Виконується перед деплоєм коду | Етап валідаціі ПЗ.  Включає в себе виконання програмного коду  Забезпечує функціональність продукту  Виконується на більш пізніх етапах розробки програмного забезпечення  Велика вартість виправлення багів  Покриває обмежену область коду, потребує меншого охоплення  Включає в себе як функціональне так і нефункціональне тестування  Мета – пошук і усунення багів  Виявляється менше дефектів,аніж при статичному тестуванні  Виконується після деплоя коду |
| **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** | Процес статичного тестування може займати багато часу, так як в основному він виконується вручну. | Оскільки динамічне тестування являє собою складний процес, воно займає багато часу. |
| **Обмеження 2** | Перешкоджає виявленню вразливостей, представлених в середовищі виконання. | Висока вартість проведення тестування. |
| **Обмеження 3** |  | 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. | 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. |

**Другий рівень**

**Завдання 2**

1. 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).*

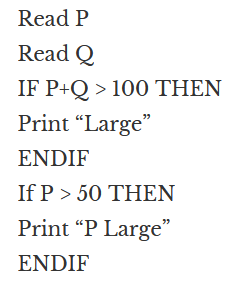
Яке твердження є коректним?

* 1. Коректно. Будь-який тест кейс надає 100% покриття тверджень, таким чином покриває 50% рішень.
  2. **Коректно. Результат будь-якого тесту умови IF буде або правдими, або ні.**
  3. Некоректно. Один тест може гарантувати 25% перевірки рішень в цьому випадку.
  4. Некоректно, бо занадто загальне твердження. Ми не можемо знати, чи є воно коректним, бо це залежить від тестованого ПЗ.

1. 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?

* 1. **1 – для покриття операторів, 2 – для покриття рішень**
  2. 1 – для покриття операторів, 1 – для покриття рішень
  3. 2 – для покриття операторів, 2 – для покриття рішень
  4. 2 – для покриття операторів, 1 – для покриття рішень

1. Скільки потрібно тестів для перевірки тверджень коду: 
   1. 2
   2. 1
   3. 3
   4. **4**

**Третій рівень**

**Завдання 2**

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»*

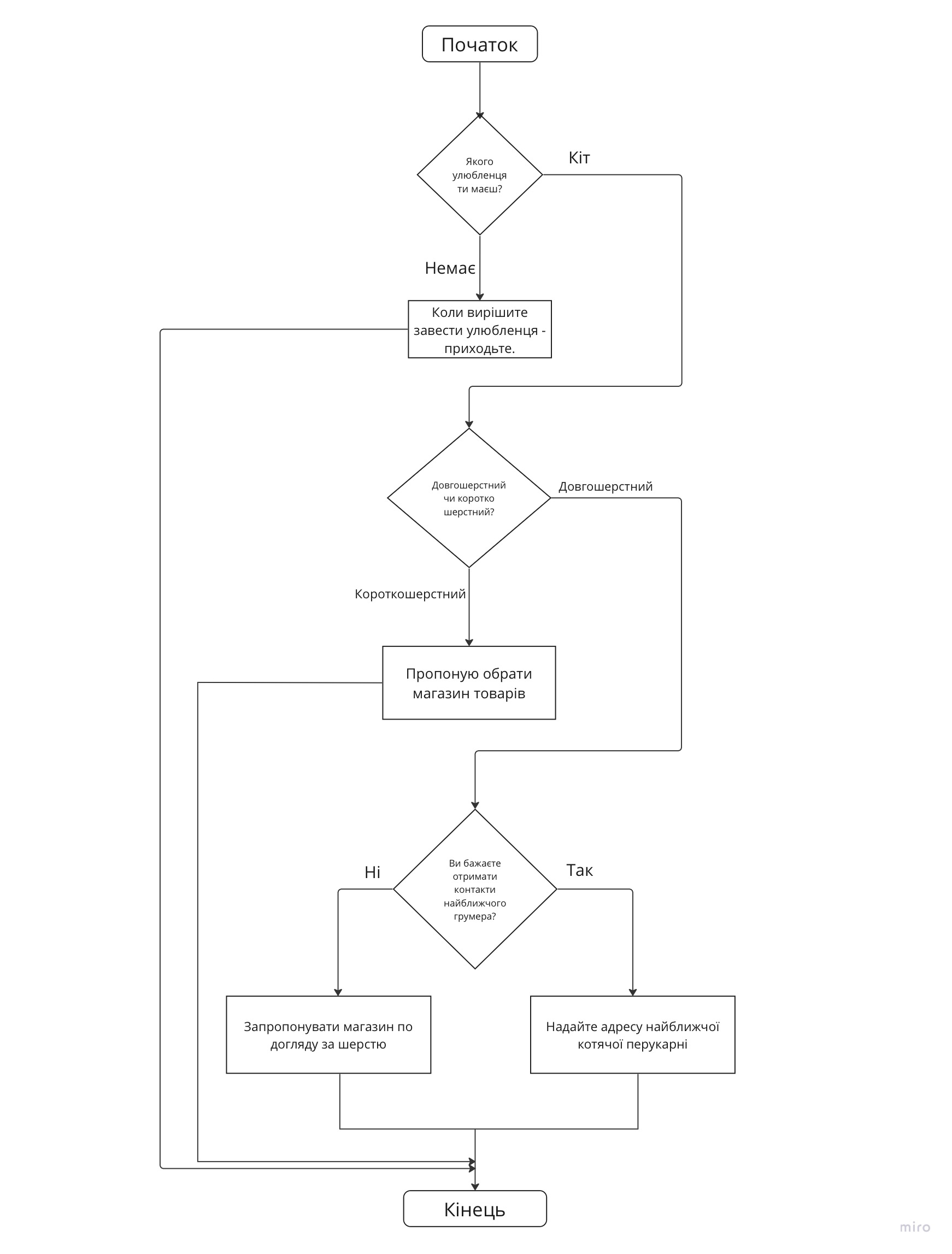
*Закінчити*

*Якщо клієнт не має кота*

*Скажи “Коли вирішите завести улюбленця – приходьте”*

*Закінчити*

1. **Намалюй схему алгоритму**



1. мінімальний набір тест-кейсів 4, щоб переконатися, що всі запитання були поставлені, всі комбінації були пройдені та всі відповіді були отримані.