1.ให้นักศึกษา สร้าง โปรแกรม ฟังก์ชัน บวก ลบ คูณ หาร โดย Copilot

```
Code  C++
#include <iostream>

using namespace std;

double add(double a, double b) {
   return a + b;
}

double subtract(double a, double b) {
   return a - b;
}

double multiply(double a, double b) {
   return a * b;
}

double divide(double a, double b) {
   if (b != 0) {
      return a / b;
   } else {
      cout << "Error: Division by zero!" << endl;
      return 0;
   }
}

int main() {
   double num1, num2;
   char operation;

   cout << "Enter first number: ";
   cin >> num1;
   cout << "Enter an operator (+, -, *, /): ";
   cin >> operation;
   cout << "Enter second number: ";
   cin >> num2;

   switch (operation) {
      case '+':
         cout << "Result: " << add(num1, num2) << endl;
         break;
      case '-':
         cout << "Result: " << subtract(num1, num2) << endl;
         break;
      case '*':
         cout << "Result: " << multiply(num1, num2) << endl;
```

```
        break;
    case '/':
        cout << "Result: " << divide(num1, num2) << endl;
        break;
    default:
        cout << "Error: Invalid operator!" << endl;
        break;
  }

  return 0;
}
```

ผลการทดลอง

```
TERMINAL    GITLENS    PROBLEMS    OUTPUT    PO

> g++ w1.cpp -o main && ./main
Enter first number: .1
Enter an operator (+, -, *, /): +
Enter second number: .2
Result: 0.3
> g++ w1.cpp -o main && ./main
Enter first number: 1234
Enter an operator (+, -, *, /): /
Enter second number: 4
Result: 308.5
> g++ w1.cpp -o main && ./main
Enter first number: 2
Enter an operator (+, -, *, /): *
Enter second number: 4
Result: 8
> g++ w1.cpp -o main && ./main
Enter first number: 9876
Enter an operator (+, -, *, /): -
Enter second number: 1234
Result: 8642
 ~/Desktop/MUT/Y_1 T_2/MIIA0106/lab11 | main
> ›
```

Code Python

```
def add(a, b):
   return a + b

def subtract(a, b):
   return a - b

def multiply(a, b):
   return a * b

def divide(a, b):
   if b != 0:
      return a / b
   else:
      print("Error: Division by zero!")
      return 0

def main():
   num1 = float(input("Enter first number: "))
   operation = input("Enter an operator (+, -, *, /): ")
   num2 = float(input("Enter second number: "))

   if operation == '+':
      print("Result:", add(num1, num2))
   elif operation == '-':
      print("Result:", subtract(num1, num2))
   elif operation == '*':
      print("Result:", multiply(num1, num2))
   elif operation == '/':
      print("Result:", divide(num1, num2))
   else:
      print("Error: Invalid operator!")

if __name__ == "__main__":
   main()
```

ผลการทดลอง

```
TERMINAL    GITLENS    PROBLEMS    OUTPUT

> python3 w1.py
Enter first number: .1
Enter an operator (+, -, *, /): +
Enter second number: .2
Result: 0.30000000000000004
> python3 w1.py
Enter first number: 99
Enter an operator (+, -, *, /): -88
Enter second number: 88
Error: Invalid operator!
> python3 w1.py
Enter first number: 99
Enter an operator (+, -, *, /): -
Enter second number: 88
Result: 11.0
```

2.ให้นักศึกษา สร้างโจทย์เกี่ยวกับ Stack (LIFO) พร้อมเขียน code C++ และ python

โจทย์ การใช้ undo ใน LIFO

Code C++

```
#include <iostream>
#include <stack>
#include <string>

class Stack {
private:
   std::stack<std::string> items;

public:
   void push(const std::string& item) {
      items.push(item);
   }

   void pop() {
      if (!items.empty()) {
         items.pop();
      }
   }

   std::string peek() const {
      if (!items.empty()) {
         return items.top();
      }
      return "";
   }

   bool is_empty() const {
      return items.empty();
   }
};

// Example usage: Undo functionality in a text editor
int main() {
   Stack editor_stack;
   editor_stack.push("Type 'Hello'");
   editor_stack.push("Type 'World'");
   editor_stack.push("Delete 'World'");
```

```
    std::cout << "Current state: " << editor_stack.peek() << std::endl;  // Output: Delete 'World'
    editor_stack.pop();
    std::cout << "Undo last action. Current state: " << editor_stack.peek() << std::endl;  // Output: Type
'World'
    editor_stack.pop();
    std::cout << "Undo last action. Current state: " << editor_stack.peek() << std::endl;  // Output: Type
'Hello'

    return 0;
}
```

ผลการทดลอง

```
TERMINAL    GITLENS    PROBLEMS    OUTPUT    POR

> g++ w2.cpp -o w2 && ./w2
Current state: Delete 'World'
Undo last action. Current state: Type 'World'
Undo last action. Current state: Type 'Hello'
```

Code Python

```python
class Stack:
    def __init__(self):
        self.items = []

    def push(self, item):
        self.items.append(item)

    def pop(self):
        if not self.is_empty():
            return self.items.pop()
        return None

    def is_empty(self):
        return len(self.items) == 0

    def peek(self):
        if not self.is_empty():
            return self.items[-1]
        return None

# Example usage: Undo functionality in a text editor
if __name__ == "__main__":
    editor_stack = Stack()
    editor_stack.push("Type 'Hello'")
    editor_stack.push("Type 'World'")
    editor_stack.push("Delete 'World'")

    print("Current state:", editor_stack.peek())  # Output: Delete 'World'
    editor_stack.pop()
    print("Undo last action. Current state:", editor_stack.peek())  # Output: Type 'World'
    editor_stack.pop()
    print("Undo last action. Current state:", editor_stack.peek())  # Output: Type 'Hello'
```

ผลการทดลอง

```
TERMINAL    GITLENS    PROBLEMS    OUTPUT    PORT

> python3 w2.py
Current state: Delete 'World'
Undo last action. Current state: Type 'World'
Undo last action. Current state: Type 'Hello'
```

3.ให้นักศึกษา สร้างโจทย์เกี่ยวกับ Queue (FIFO) พร้อมเขียน code C++ และ python

โจทย์

สร้าง FIFO สำหรับจัดการTicketing system

Code C++

```
#include <iostream>
#include <queue>
#include <string>

class Queue {
private:
   std::queue<std::string> items;

public:
   bool is_empty() const {
      return items.empty();
   }

   void enqueue(const std::string& item) {
      items.push(item);
   }

   void dequeue() {
      if (!is_empty()) {
         items.pop();
      } else {
         throw std::out_of_range("dequeue from empty queue");
      }
   }

   size_t size() const {
      return items.size();
   }

   std::string front() const {
      if (!is_empty()) {
         return items.front();
      } else {
         throw std::out_of_range("front from empty queue");
      }
   }
};
```

```
// Real-world example: Ticketing system
int main() {
    Queue ticket_queue;

    // Customers arriving at the ticket counter
    ticket_queue.enqueue("Customer 1");
    ticket_queue.enqueue("Customer 2");
    ticket_queue.enqueue("Customer 3");

    std::cout << "Queue size: " << ticket_queue.size() << std::endl;

    // Serving customers
    std::cout << "Serving: " << ticket_queue.front() << std::endl;
    ticket_queue.dequeue();
    std::cout << "Serving: " << ticket_queue.front() << std::endl;
    ticket_queue.dequeue();
    std::cout << "Queue size: " << ticket_queue.size() << std::endl;

    return 0;
}
```

ผลการทดลอง

```
TERMINAL    GITLENS    PROBLEMS

> g++ w3.cpp -o w3 && ./w3
Queue size: 3
Serving: Customer 1
Serving: Customer 2
Queue size: 1
 ~/Desktop/MUT/Y_1 T_2/MIIA0106
> 
```

Code Python

```python
class Queue:
    def __init__(self):
        self.items = []

    def is_empty(self):
        return len(self.items) == 0

    def enqueue(self, item):
        self.items.append(item)

    def dequeue(self):
        if not self.is_empty():
            return self.items.pop(0)
        else:
            raise IndexError("dequeue from empty queue")

    def size(self):
        return len(self.items)

# Real-world example: Ticketing system
ticket_queue = Queue()

# Customers arriving at the ticket counter
ticket_queue.enqueue("Customer 1")
ticket_queue.enqueue("Customer 2")
ticket_queue.enqueue("Customer 3")

print("Queue size:", ticket_queue.size())

# Serving customers
print("Serving:", ticket_queue.dequeue())
print("Serving:", ticket_queue.dequeue())
print("Queue size:", ticket_queue.size())
```

ผลการทดลอง

```
TERMINAL    GITLENS    PROBLEMS    OUTPU

> python3 w3.py
Queue size: 3
Serving: Customer 1
Serving: Customer 2
Queue size: 1
 ~/Desktop/MUT/Y_1 T_2/MIIA0106/lab11
>
```