**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

|  |
| --- |
|  |

|  |
| --- |
| 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

|  |
| --- |
|  |

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

|  |
| --- |
|  |

|  |
| --- |
| Code 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

|  |
| --- |
|  |

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

|  |
| --- |
|  |

|  |
| --- |
| Code 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

|  |
| --- |
|  |