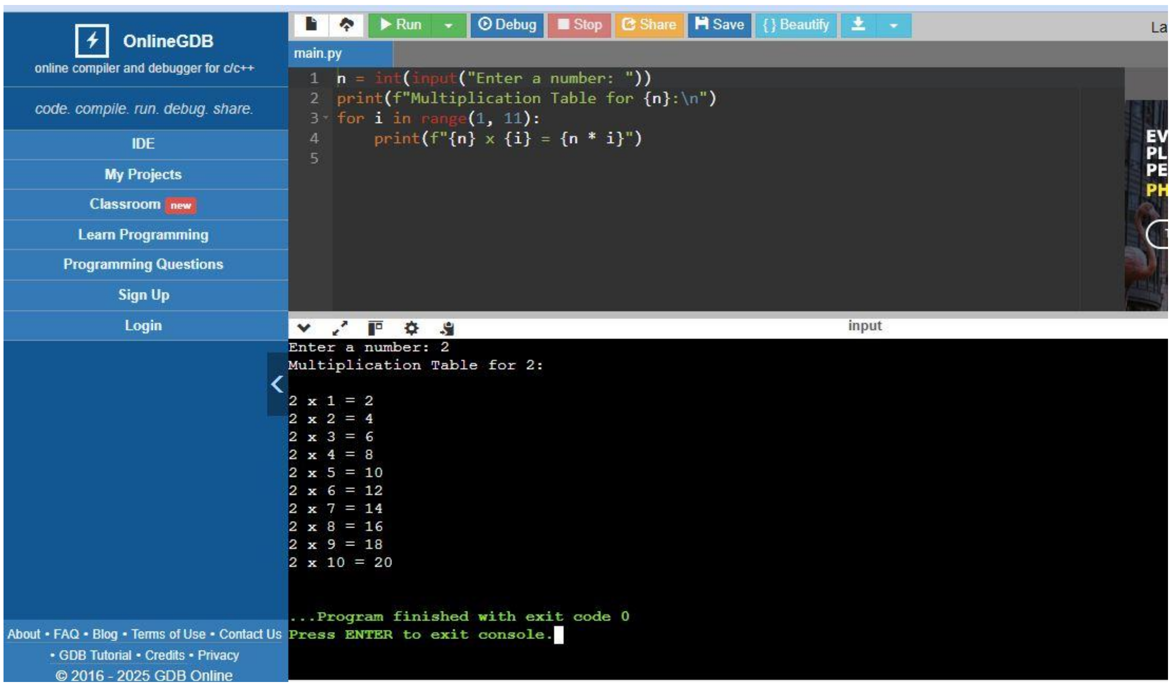
PYTHON DEVELOPER

TASK - 3

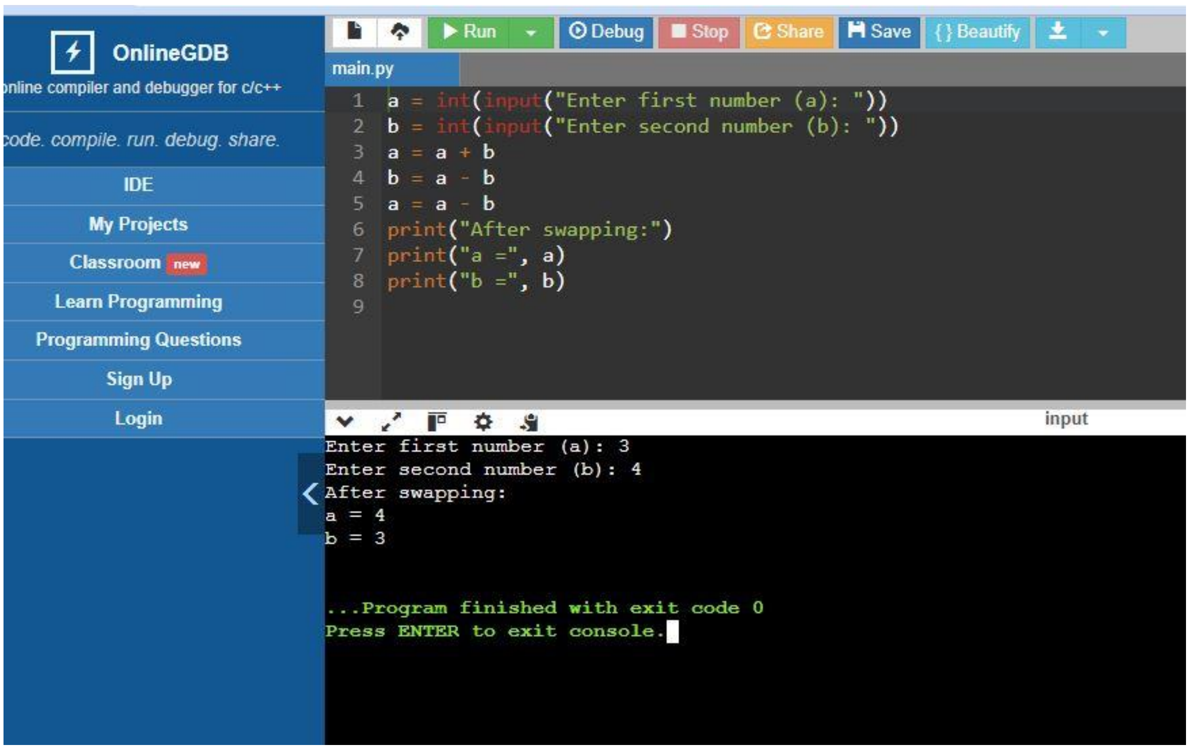
17. Table of a Number:



```python
n = int(input("Enter a number: "))
print(f"Multiplication Table for {n}:\n")
for i in range(1, 11):
    print(f"{n} x {i} = {n * i}")
```

```
Enter a number: 2
Multiplication Table for 2:

2 x 1 = 2
2 x 2 = 4
2 x 3 = 6
2 x 4 = 8
2 x 5 = 10
2 x 6 = 12
2 x 7 = 14
2 x 8 = 16
2 x 9 = 18
2 x 10 = 20


...Program finished with exit code 0
Press ENTER to exit console.
```

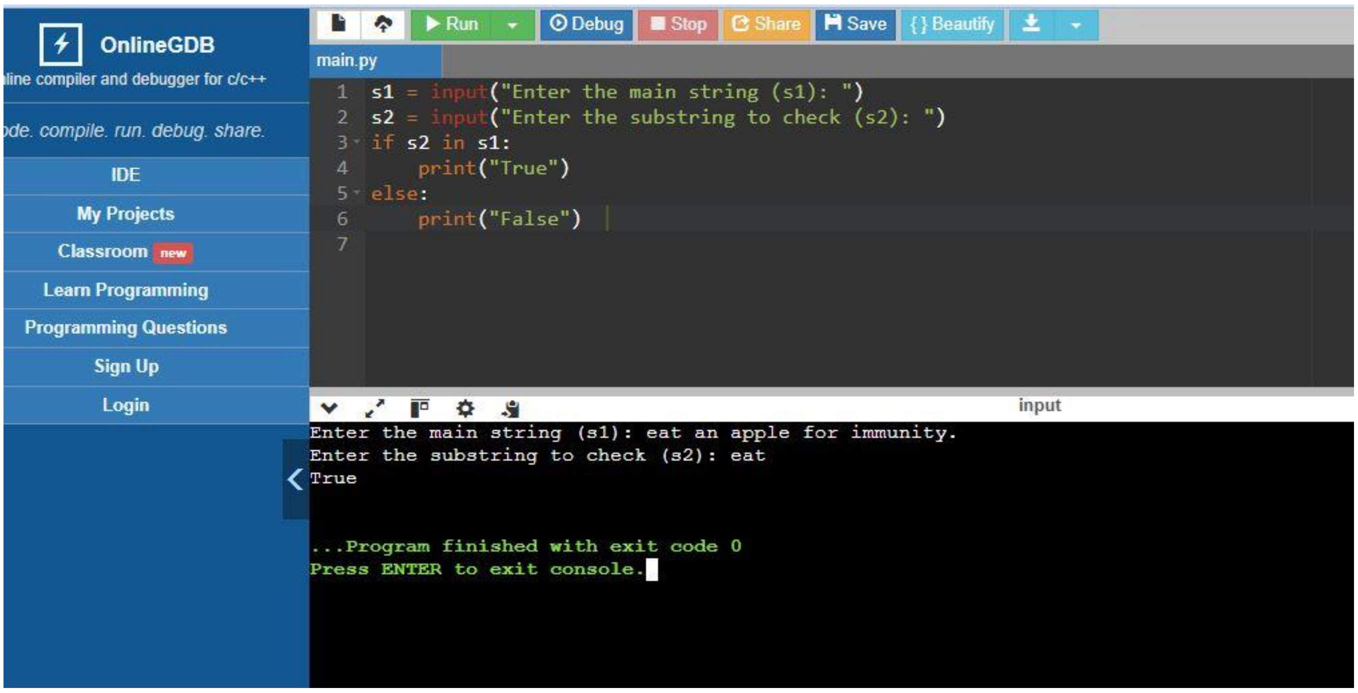
18. Swap Two Numbers:



```python
a = int(input("Enter first number (a): "))
b = int(input("Enter second number (b): "))
a = a + b
b = a - b
a = a - b
print("After swapping:")
print("a =", a)
print("b =", b)
```

```
Enter first number (a): 3
Enter second number (b): 4
After swapping:
a = 4
b = 3


...Program finished with exit code 0
Press ENTER to exit console.
```

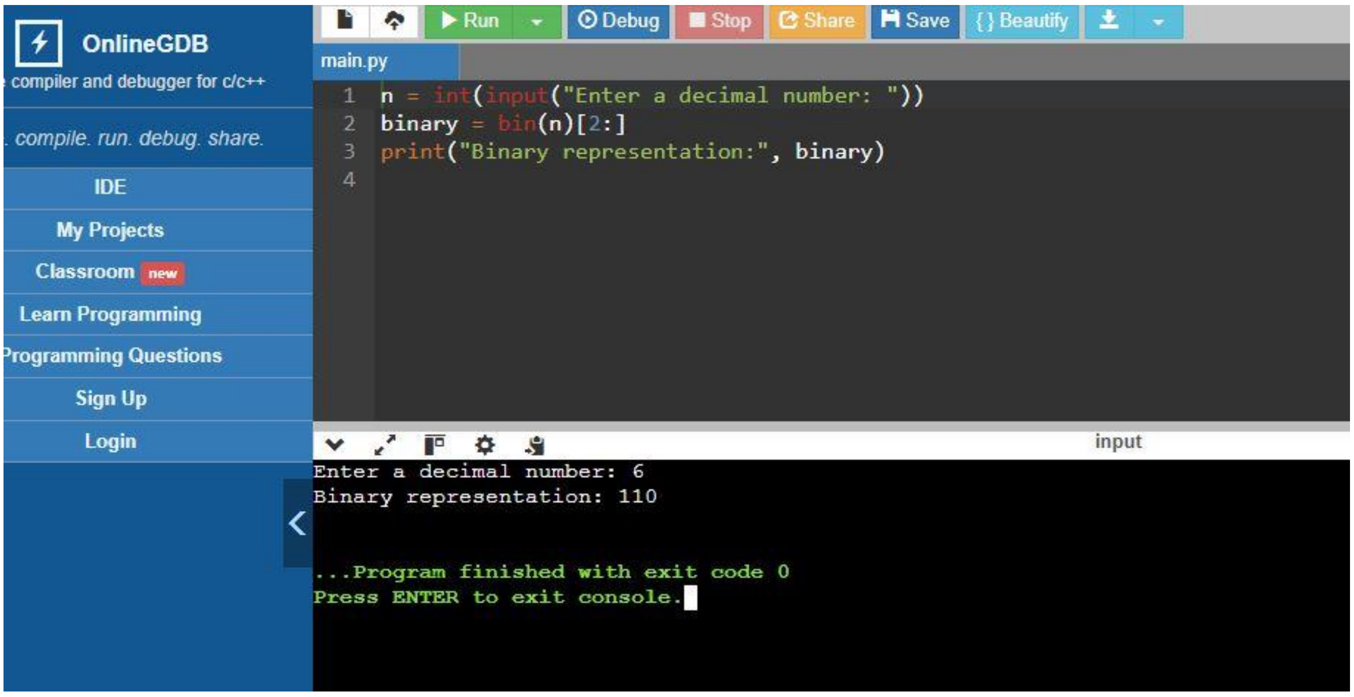
19. Check Substring:



```python
s1 = input("Enter the main string (s1): ")
s2 = input("Enter the substring to check (s2): ")
if s2 in s1:
    print("True")
else:
    print("False")
```

```
Enter the main string (s1): eat an apple for immunity.
Enter the substring to check (s2): eat
True


...Program finished with exit code 0
Press ENTER to exit console.
```

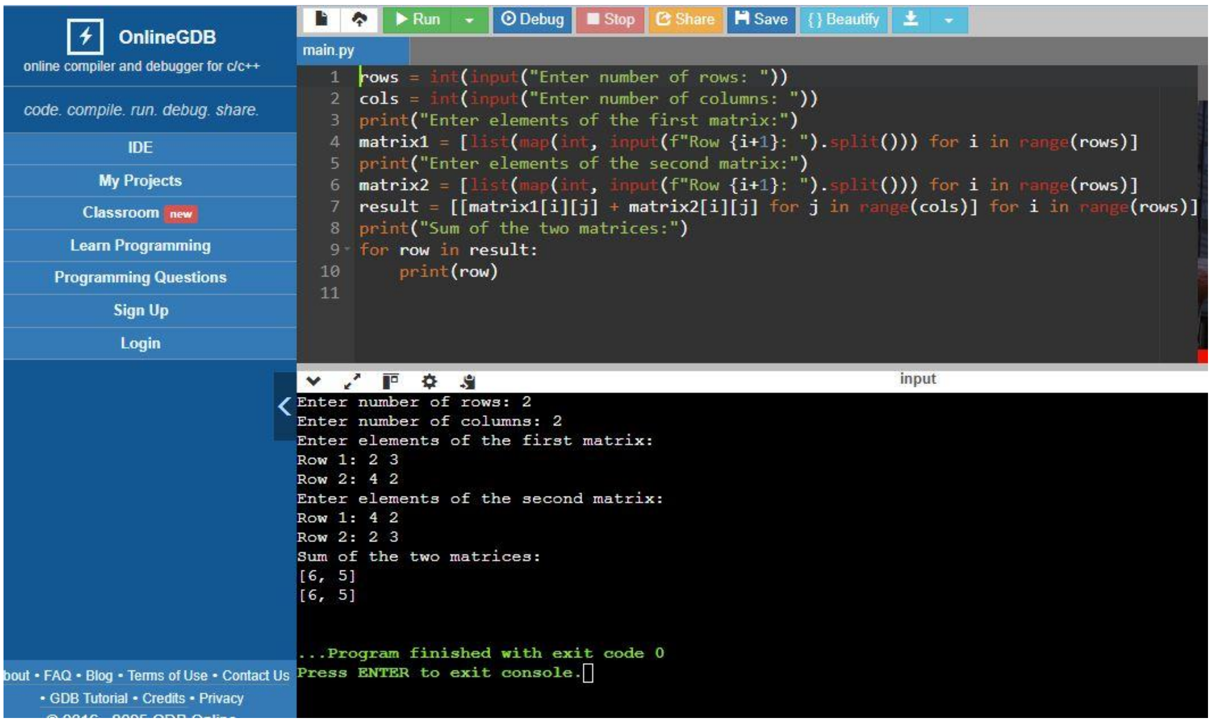
20. Decimal to Binary:



```python
n = int(input("Enter a decimal number: "))
binary = bin(n)[2:]
print("Binary representation:", binary)
```

```
Enter a decimal number: 6
Binary representation: 110


...Program finished with exit code 0
Press ENTER to exit console.
```

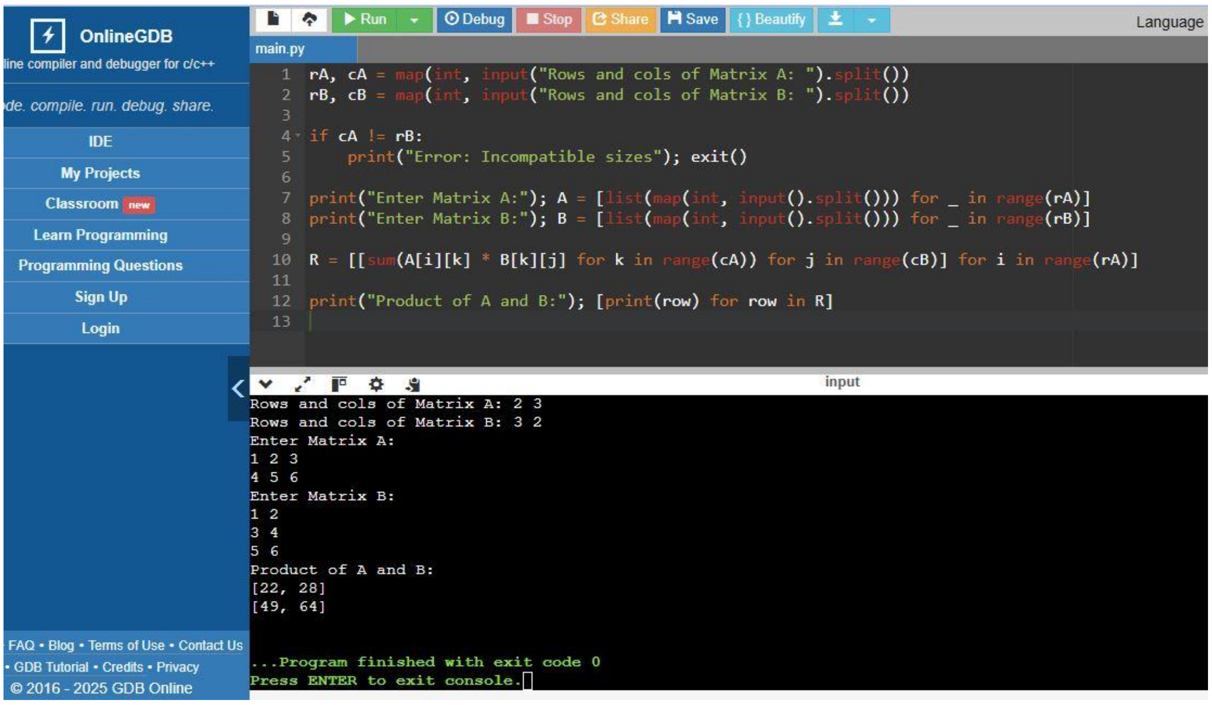
21. Matrix Addition:



```python
rows = int(input("Enter number of rows: "))
cols = int(input("Enter number of columns: "))
print("Enter elements of the first matrix:")
matrix1 = [list(map(int, input(f"Row {i+1}: ").split())) for i in range(rows)]
print("Enter elements of the second matrix:")
matrix2 = [list(map(int, input(f"Row {i+1}: ").split())) for i in range(rows)]
result = [[matrix1[i][j] + matrix2[i][j] for j in range(cols)] for i in range(rows)]
print("Sum of the two matrices:")
for row in result:
    print(row)
```

```
Enter number of rows: 2
Enter number of columns: 2
Enter elements of the first matrix:
Row 1: 2 3
Row 2: 4 2
Enter elements of the second matrix:
Row 1: 4 2
Row 2: 2 3
Sum of the two matrices:
[6, 5]
[6, 5]


...Program finished with exit code 0
Press ENTER to exit console.
```

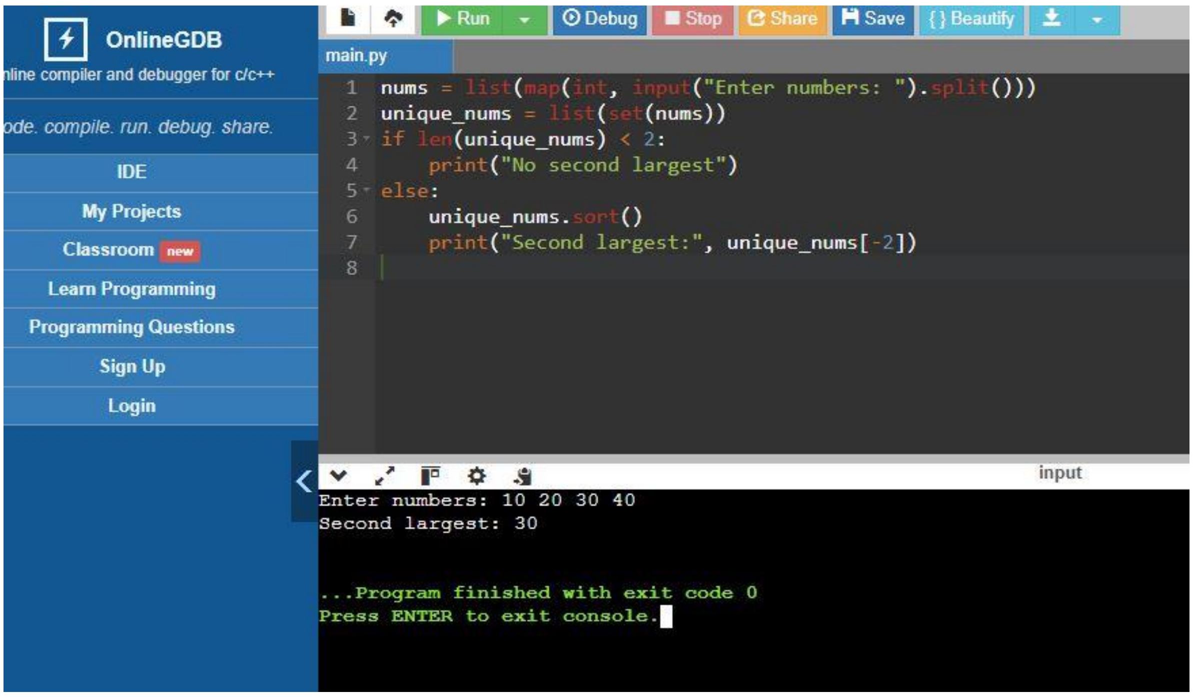
22. Matrix Multiplication:



```python
rA, cA = map(int, input("Rows and cols of Matrix A: ").split())
rB, cB = map(int, input("Rows and cols of Matrix B: ").split())

if cA != rB:
    print("Error: Incompatible sizes"); exit()

print("Enter Matrix A:"); A = [list(map(int, input().split())) for _ in range(rA)]
print("Enter Matrix B:"); B = [list(map(int, input().split())) for _ in range(rB)]

R = [[sum(A[i][k] * B[k][j] for k in range(cA)) for j in range(cB)] for i in range(rA)]

print("Product of A and B:"); [print(row) for row in R]
```

```
Rows and cols of Matrix A: 2 3
Rows and cols of Matrix B: 3 2
Enter Matrix A:
1 2 3
4 5 6
Enter Matrix B:
1 2
3 4
5 6
Product of A and B:
[22, 28]
[49, 64]


...Program finished with exit code 0
Press ENTER to exit console.
```

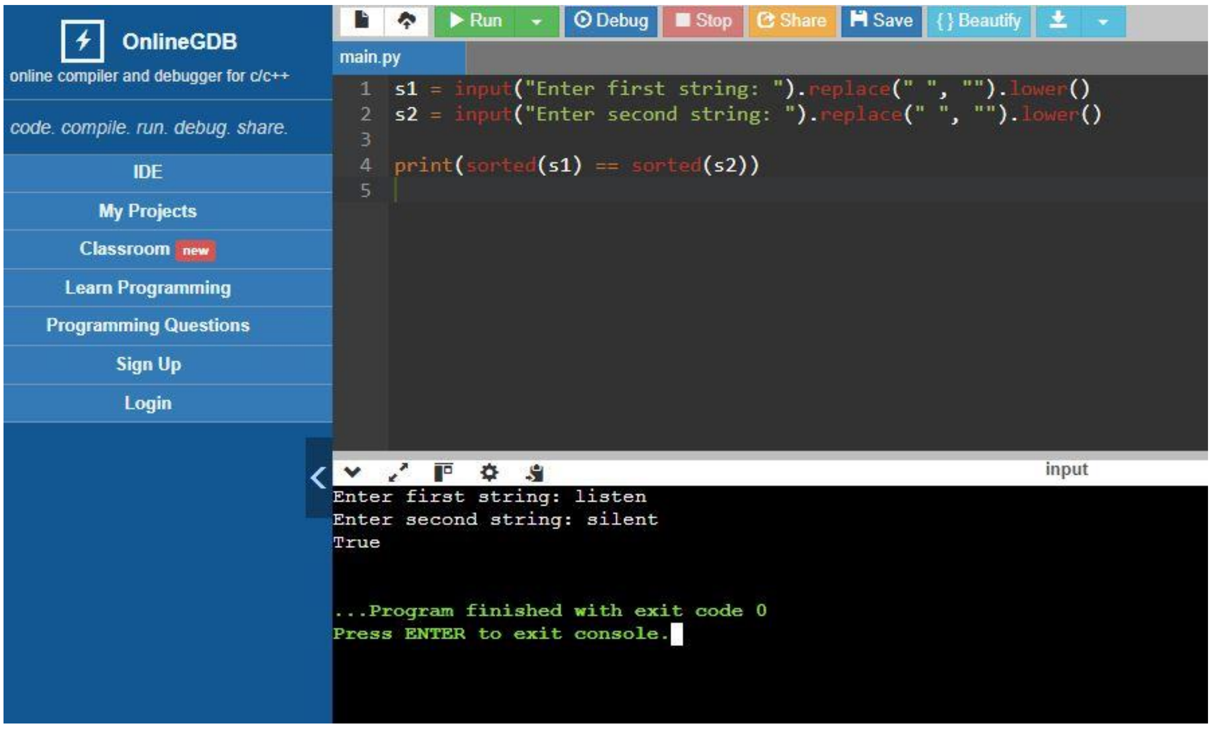
23. Find Second Larger:



```python
nums = list(map(int, input("Enter numbers: ").split()))
unique_nums = list(set(nums))
if len(unique_nums) < 2:
    print("No second largest")
else:
    unique_nums.sort()
    print("Second largest:", unique_nums[-2])
```

```
Enter numbers: 10 20 30 40
Second largest: 30


...Program finished with exit code 0
Press ENTER to exit console.
```

24. Check Anagram:



```python
s1 = input("Enter first string: ").replace(" ", "").lower()
s2 = input("Enter second string: ").replace(" ", "").lower()

print(sorted(s1) == sorted(s2))
```

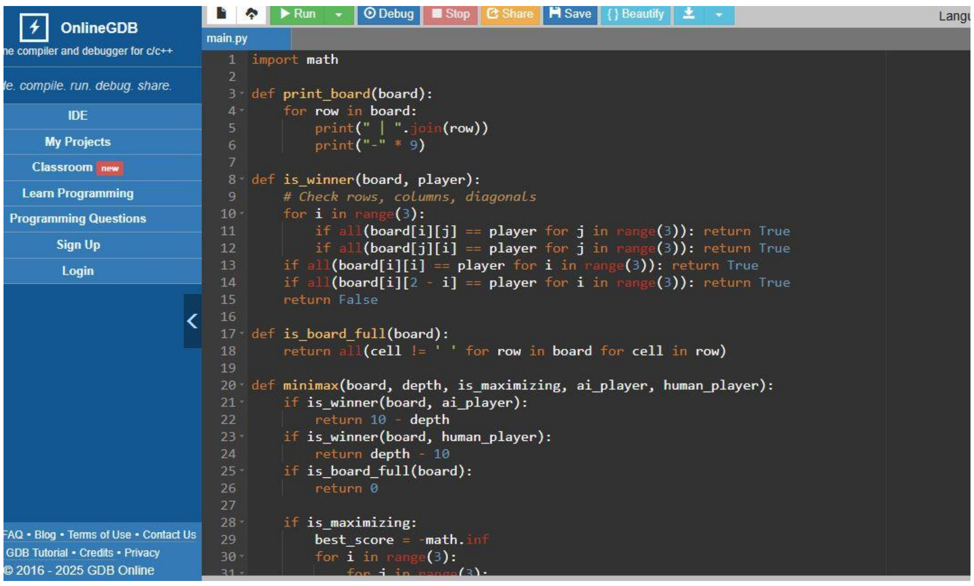
```
Enter first string: listen
Enter second string: silent
True


...Program finished with exit code 0
Press ENTER to exit console.
```

3. AI-Based Tic-Tac-Toe

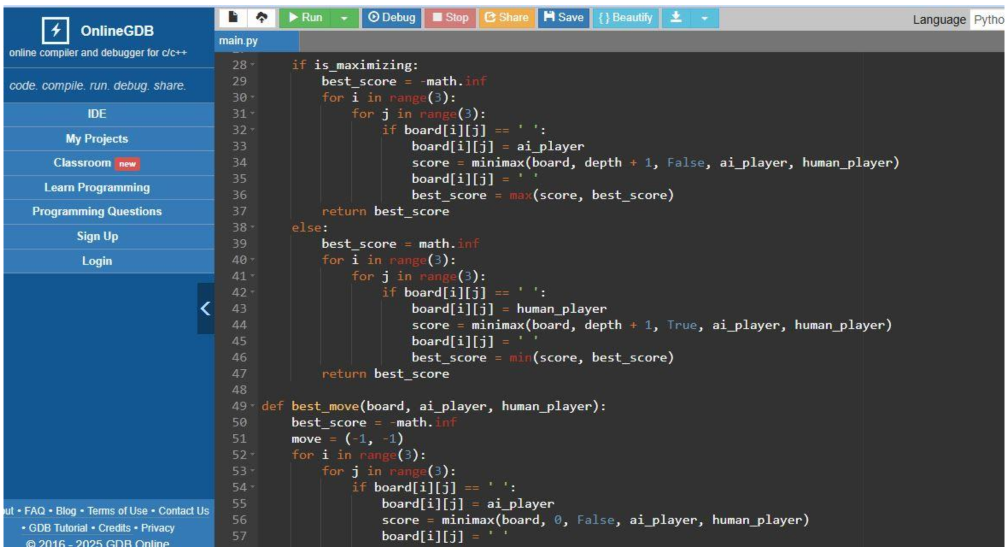
● Description: Create a Tic-Tac-Toe game where the computer plays against the user anduses a minimax algorithm to make decisions.

```
import math


def print_board(board):
    for row in board:
        print(" | ".join(row))
        print("-" * 9)


def is_winner(board, player):
    # Check rows, columns, diagonals
    for i in range(3):
        if all(board[i][j] == player for j in range(3)): return True
        if all(board[j][i] == player for j in range(3)): return True
    if all(board[i][i] == player for i in range(3)): return True
    if all(board[i][2 - i] == player for i in range(3)): return True
    return False


def is_board_full(board):
    return all(cell != ' ' for row in board for cell in row)


def minimax(board, depth, is_maximizing, ai_player, human_player):
    if is_winner(board, ai_player):
        return 10 - depth
    if is_winner(board, human_player):
        return depth - 10
    if is_board_full(board):
        return 0
    if is_maximizing:
        best_score = -math.inf
        for i in range(3):
            for j in range(3):
                if board[i][j] == ' ':
                    board[i][j] = ai_player
                    score = minimax(board, depth + 1, False, ai_player, human_player)
                    board[i][j] = ' '
                    best_score = max(score, best_score)
        return best_score
    else:
        best_score = math.inf
        for i in range(3):
            for j in range(3):
                if board[i][j] == ' ':
                    board[i][j] = human_player
                    score = minimax(board, depth + 1, True, ai_player, human_player)
                    board[i][j] = ' '
                    best_score = min(score, best_score)
        return best_score


def best_move(board, ai_player, human_player):
    best_score = -math.inf
    move = (-1, -1)
    for i in range(3):
        for j in range(3):
            if board[i][j] == ' ':
                board[i][j] = ai_player
                score = minimax(board, 0, False, ai_player, human_player)
                board[i][j] = ' '
                if score > best_score:
                    best_score = score
                    move = (i, j)
    return move


def play_game():
    board = [[' ' for _ in range(3)] for _ in range(3)]
    human_player = ''
    ai_player = ''


    while human_player not in ['X', 'O']:
        human_player = input("Choose your symbol (X/O): ").upper()
    ai_player = 'O' if human_player == 'X' else 'X'


    current_turn = 'X'  # X always starts
    print("\nBoard positions are numbered 1-9 as below:")
    print("1 | 2 | 3\n4 | 5 | 6\n7 | 8 | 9\n")


    while True:
        print_board(board)


        if current_turn == human_player:
            valid_move = False
            while not valid_move:
                try:
                    move = int(input("Your move (1-9): ")) - 1
                    row, col = divmod(move, 3)
                    if board[row][col] == ' ':
                        board[row][col] = human_player
                        valid_move = True
                    else:
                        print("Cell occupied! Try again.")
                except (ValueError, IndexError):
                    print("Invalid input! Enter a number from 1 to 9.")

        else:
            print("AI is making a move...")
            row, col = best_move(board, ai_player, human_player)
            board[row][col] = ai_player

        # Check for win/tie
        if is_winner(board, current_turn):
            print_board(board)
            if current_turn == human_player:
                print("Congratulations! You won!")
            else:
                print("AI wins! Better luck next time.")
            break

        if is_board_full(board):
            print_board(board)
            print("It's a tie!")
            break

        # Switch turns
        current_turn = ai_player if current_turn == human_player else human_player

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

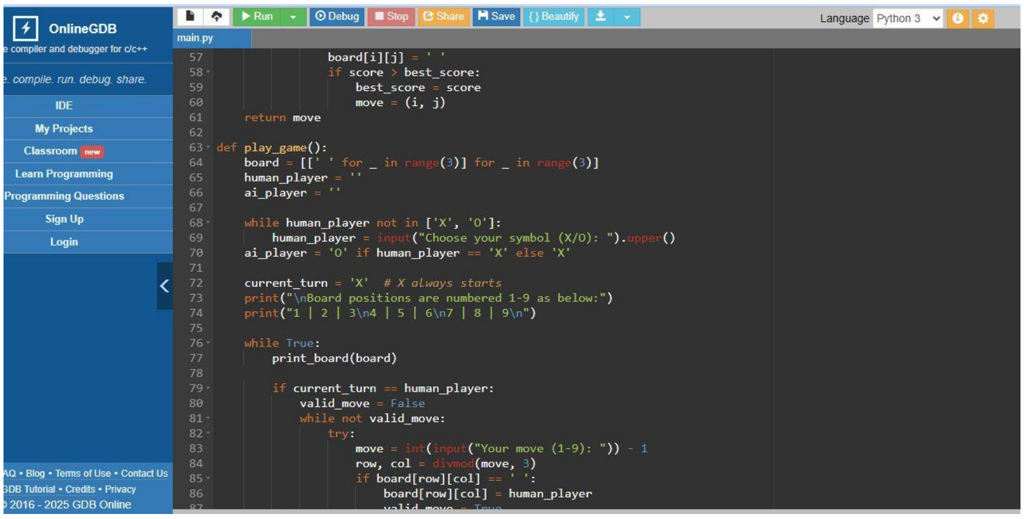
program:



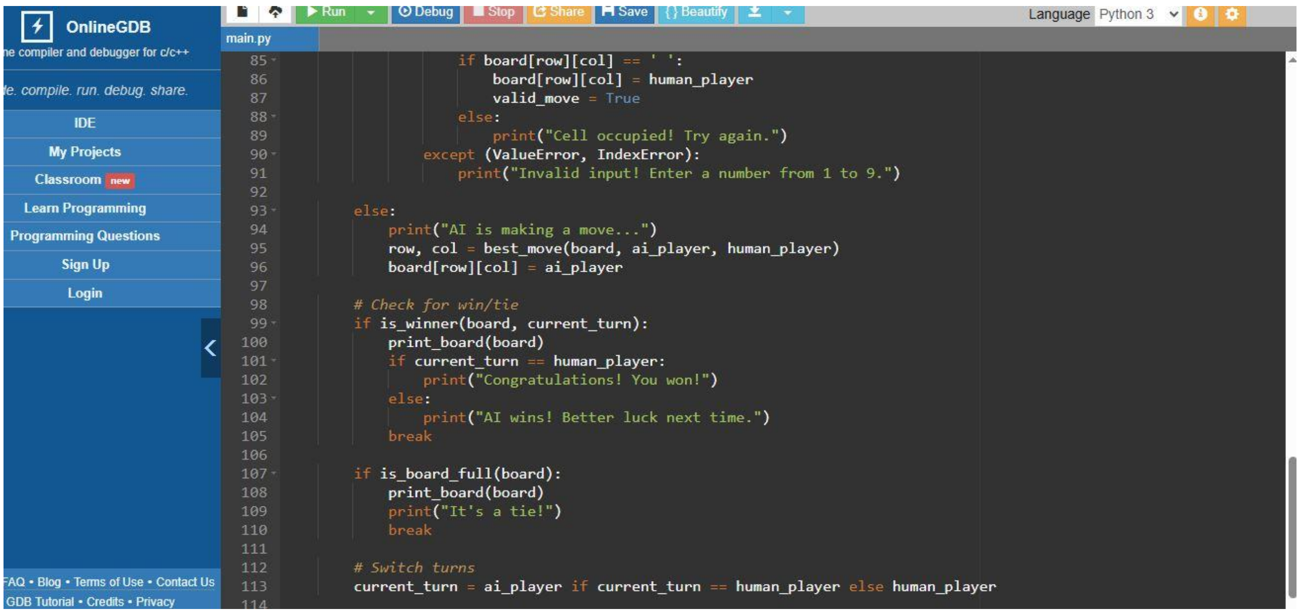
```python
import math

def print_board(board):
    for row in board:
        print(" | ".join(row))
        print("-" * 9)

def is_winner(board, player):
    # Check rows, columns, diagonals
    for i in range(3):
        if all(board[i][j] == player for j in range(3)): return True
        if all(board[j][i] == player for j in range(3)): return True
    if all(board[i][i] == player for i in range(3)): return True
    if all(board[i][2 - i] == player for i in range(3)): return True
    return False

def is_board_full(board):
    return all(cell != ' ' for row in board for cell in row)

def minimax(board, depth, is_maximizing, ai_player, human_player):
    if is_winner(board, ai_player):
        return 10 - depth
    if is_winner(board, human_player):
        return depth - 10
    if is_board_full(board):
        return 0

    if is_maximizing:
        best_score = -math.inf
        for i in range(3):
            for j in range(3):
```



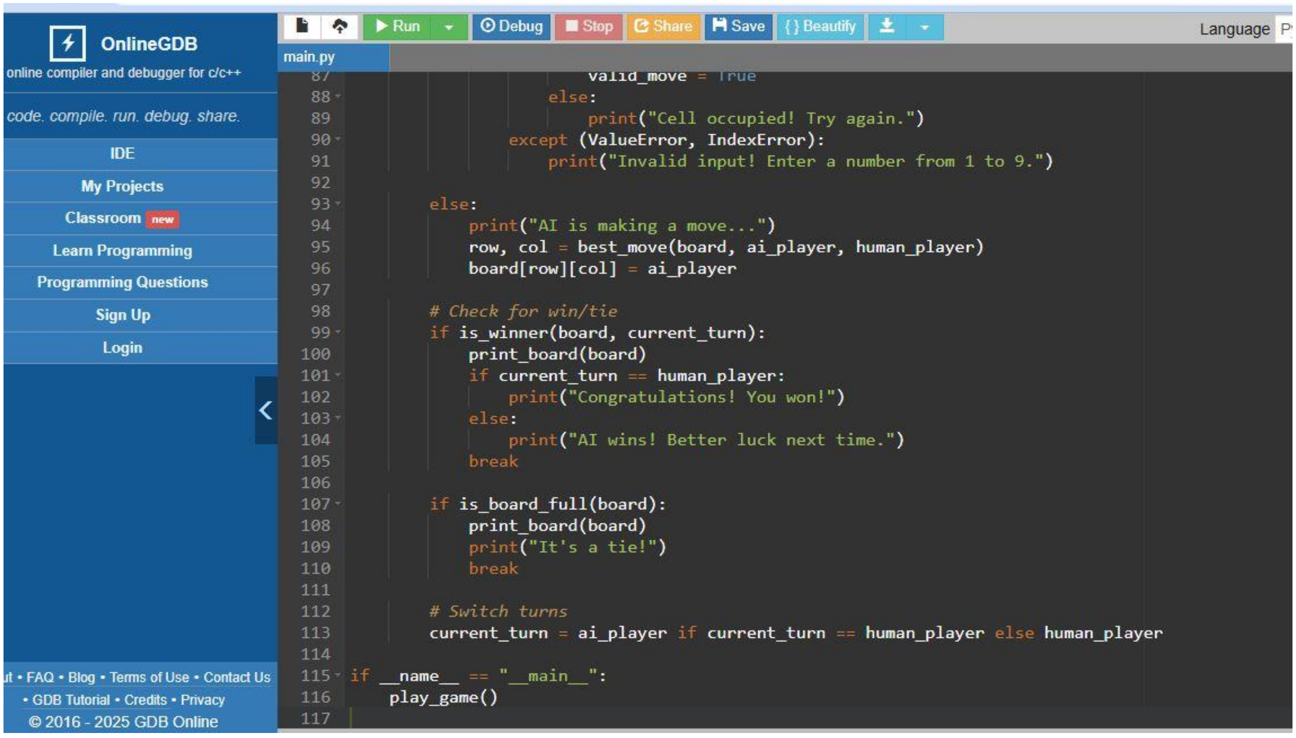
```python
    if is_maximizing:
        best_score = -math.inf
        for i in range(3):
            for j in range(3):
                if board[i][j] == ' ':
                    board[i][j] = ai_player
                    score = minimax(board, depth + 1, False, ai_player, human_player)
                    board[i][j] = ' '
                    best_score = max(score, best_score)
        return best_score
    else:
        best_score = math.inf
        for i in range(3):
            for j in range(3):
                if board[i][j] == ' ':
                    board[i][j] = human_player
                    score = minimax(board, depth + 1, True, ai_player, human_player)
                    board[i][j] = ' '
                    best_score = min(score, best_score)
        return best_score

def best_move(board, ai_player, human_player):
    best_score = -math.inf
    move = (-1, -1)
    for i in range(3):
        for j in range(3):
            if board[i][j] == ' ':
                board[i][j] = ai_player
                score = minimax(board, 0, False, ai_player, human_player)
                board[i][j] = ' '
```



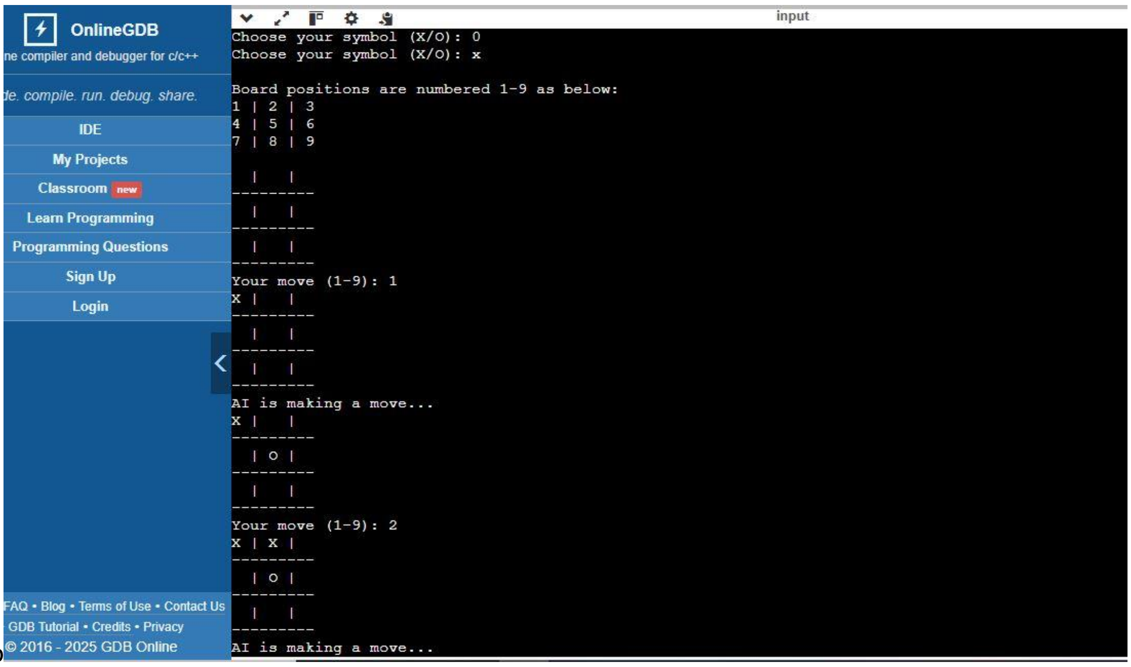
```python
                board[i][j] = ' '
                if score > best_score:
                    best_score = score
                    move = (i, j)
    return move

def play_game():
    board = [[' ' for _ in range(3)] for _ in range(3)]
    human_player = ''
    ai_player = ''

    while human_player not in ['X', 'O']:
        human_player = input("Choose your symbol (X/O): ").upper()
    ai_player = 'O' if human_player == 'X' else 'X'

    current_turn = 'X'  # X always starts
    print("\nBoard positions are numbered 1-9 as below:")
    print("1 | 2 | 3\n4 | 5 | 6\n7 | 8 | 9\n")

    while True:
        print_board(board)

        if current_turn == human_player:
            valid_move = False
            while not valid_move:
                try:
                    move = int(input("Your move (1-9): ")) - 1
                    row, col = divmod(move, 3)
                    if board[row][col] == ' ':
                        board[row][col] = human_player
                        valid_move = True
```



```python
                    if board[row][col] == ' ':
                        board[row][col] = human_player
                        valid_move = True
                    else:
                        print("Cell occupied! Try again.")
                except (ValueError, IndexError):
                    print("Invalid input! Enter a number from 1 to 9.")

        else:
            print("AI is making a move...")
            row, col = best_move(board, ai_player, human_player)
            board[row][col] = ai_player

        # Check for win/tie
        if is_winner(board, current_turn):
            print_board(board)
            if current_turn == human_player:
                print("Congratulations! You won!")
            else:
                print("AI wins! Better luck next time.")
            break

        if is_board_full(board):
            print_board(board)
            print("It's a tie!")
            break

        # Switch turns
        current_turn = ai_player if current_turn == human_player else human_player
```

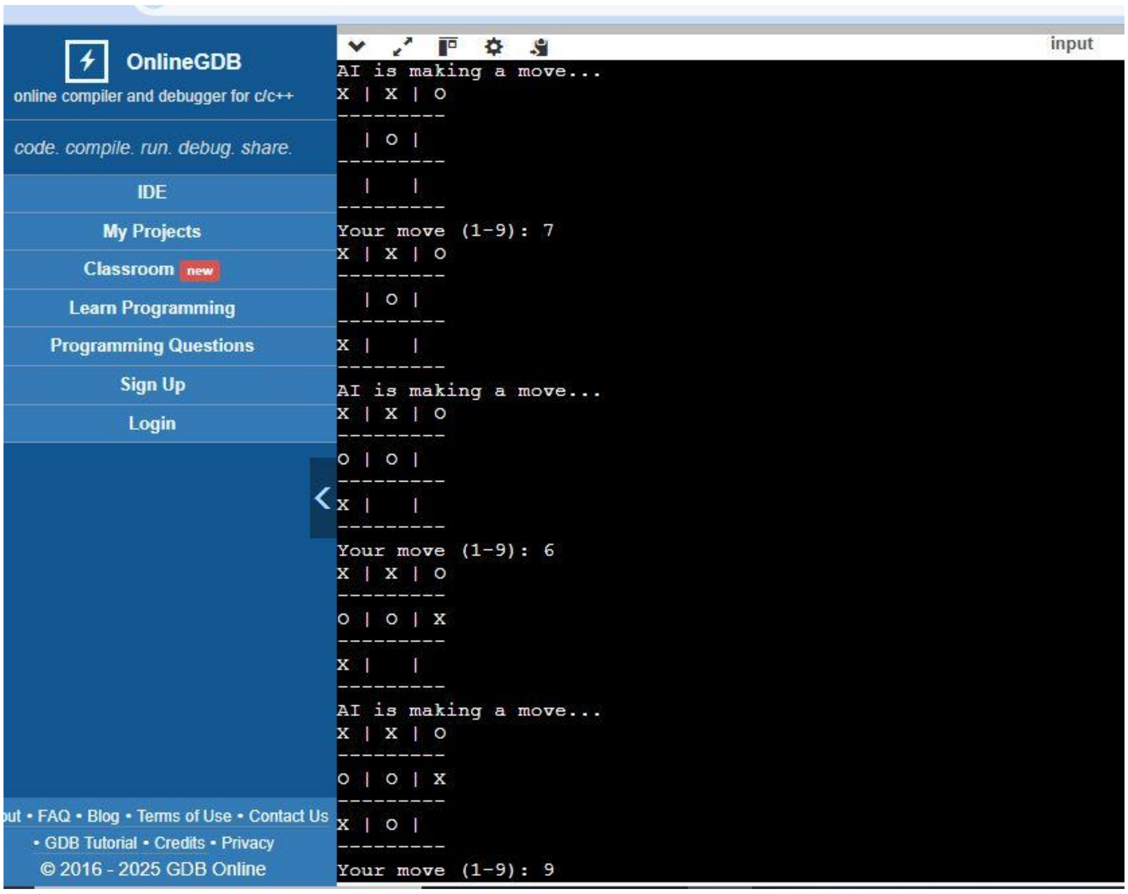


```python
                        valid_move = True
                    else:
                        print("Cell occupied! Try again.")
                except (ValueError, IndexError):
                    print("Invalid input! Enter a number from 1 to 9.")

        else:
            print("AI is making a move...")
            row, col = best_move(board, ai_player, human_player)
            board[row][col] = ai_player

        # Check for win/tie
        if is_winner(board, current_turn):
            print_board(board)
            if current_turn == human_player:
                print("Congratulations! You won!")
            else:
                print("AI wins! Better luck next time.")
            break

        if is_board_full(board):
            print_board(board)
            print("It's a tie!")
            break

        # Switch turns
        current_turn = ai_player if current_turn == human_player else human_player

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

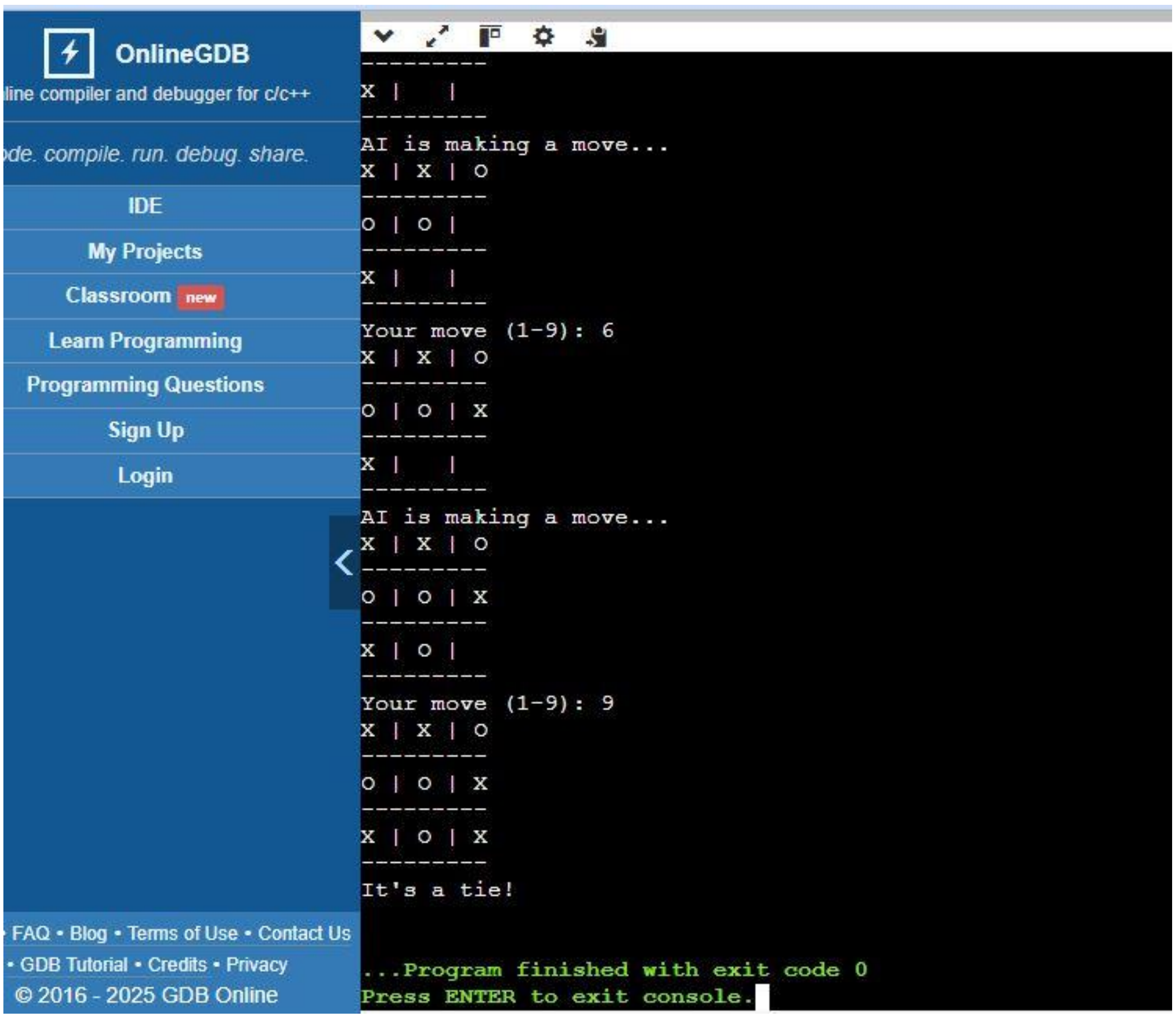

OnlineGDB
ne compiler and debugger for c/c++
de. compile. run. debug. share.
IDE
My Projects
Classroom new
Learn Programming
Programming Questions
Sign Up
Login
FAQ • Blog • Terms of Use • Contact Us
GDB Tutorial • Credits • Privacy
© 2016 - 2025 GDB Online
input

```
Choose your symbol (X/O): 0
Choose your symbol (X/O): x

Board positions are numbered 1-9 as below:
1 | 2 | 3
4 | 5 | 6
7 | 8 | 9

  |   |
---------
  |   |
---------
  |   |
---------
Your move (1-9): 1
X |   |
---------
  |   |
---------
  |   |
---------
AI is making a move...
X |   |
---------
  | O |
---------
  |   |
---------
Your move (1-9): 2
X | X |
---------
  | O |
---------
  |   |
---------
AI is making a move...
```

oo


OnlineGDB
online compiler and debugger for c/c++
code. compile. run. debug. share.
IDE
My Projects
Classroom new
Learn Programming
Programming Questions
Sign Up
Login
ut • FAQ • Blog • Terms of Use • Contact Us
• GDB Tutorial • Credits • Privacy
© 2016 - 2025 GDB Online
input

```
AI is making a move...
X | X | O
---------
  | O |
---------
  |   |
---------
Your move (1-9): 7
X | X | O
---------
  | O |
---------
X |   |
---------
AI is making a move...
X | X | O
---------
O | O |
---------
X |   |
---------
Your move (1-9): 6
X | X | O
---------
O | O | X
---------
X |   |
---------
AI is making a move...
X | X | O
---------
O | O | X
---------
X | O |
---------
Your move (1-9): 9
```


OnlineGDB
line compiler and debugger for c/c++
ode. compile. run. debug. share.
IDE
My Projects
Classroom new
Learn Programming
Programming Questions
Sign Up
Login
FAQ • Blog • Terms of Use • Contact Us
• GDB Tutorial • Credits • Privacy
© 2016 - 2025 GDB Online

```
---------
X |   |
---------
AI is making a move...
X | X | O
---------
O | O |
---------
X |   |
---------
Your move (1-9): 6
X | X | O
---------
O | O | X
---------
X |   |
---------
AI is making a move...
X | X | O
---------
O | O | X
---------
X | O |
---------
Your move (1-9): 9
X | X | O
---------
O | O | X
---------
X | O | X
---------
It's a tie!


...Program finished with exit code 0
Press ENTER to exit console.
```