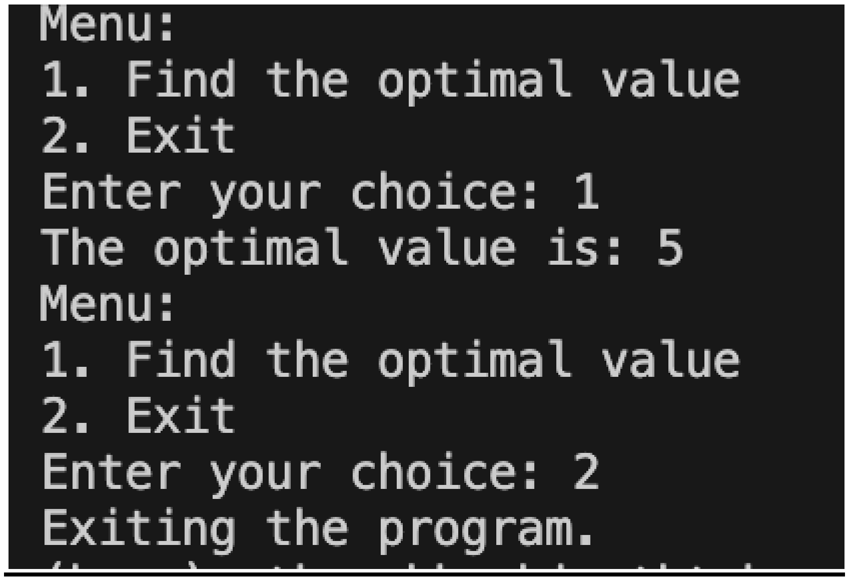
Experiment No.3:

Implement Alpha-Beta Tree search for any game search problem

Code:

```
MAX, MIN = 1000, -1000


def minimax(depth, nodeIndex, maximizingPlayer, values, alpha, beta):
    if depth == 3:
        return values[nodeIndex]


    if maximizingPlayer:
        best = MIN
        for i in range(0, 2):
            val = minimax(depth + 1, nodeIndex * 2 + i, False, values, alpha, beta)
            best = max(best, val)
            alpha = max(alpha, best)
            if beta <= alpha:
                break
        return best


    else:
        best = MAX
        for i in range(0, 2):
            val = minimax(depth + 1, nodeIndex * 2 + i, True, values, alpha, beta)
            best = min(best, val)
            beta = min(beta, best)
            if beta <= alpha:
                break
        return best


def main():
    values = [3, 5, 6, 9, 1, 2, 0, -1]


    while True:
        print("Menu:")
        print("1. Find the optimal value")
        print("2. Exit")
        choice = int(input("Enter your choice: "))


        if choice == 1:
            print("The optimal value is:", minimax(0, 0, True, values, MIN, MAX))
        elif choice == 2:
            print("Exiting the program.")
            break
        else:
            print("Invalid choice. Please enter a valid option.")


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

Output:



```
Menu:
1. Find the optimal value
2. Exit
Enter your choice: 1
The optimal value is: 5
Menu:
1. Find the optimal value
2. Exit
Enter your choice: 2
Exiting the program.
```