

Name: Md Jubair Pantho

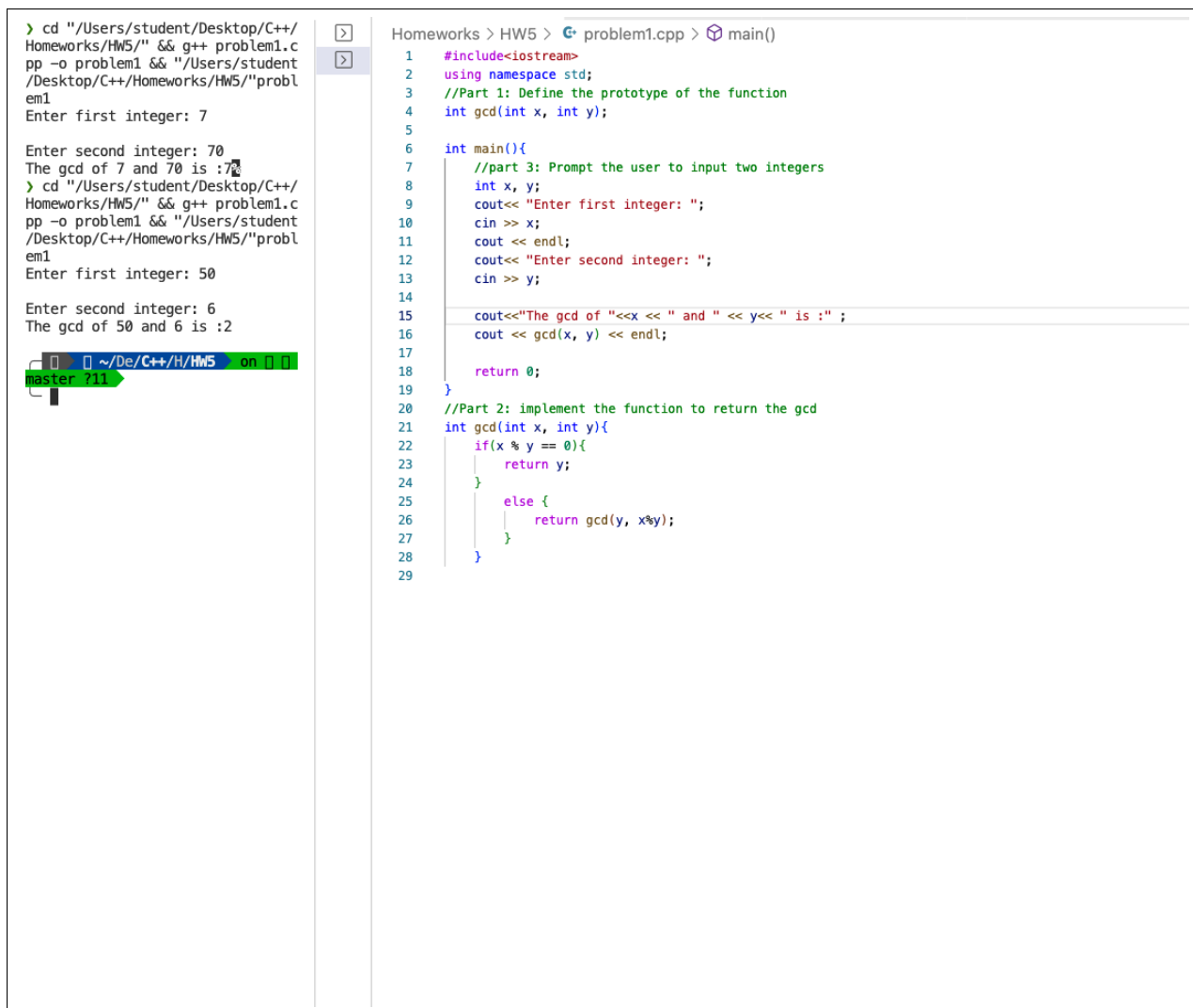
UTC ID : WZS444

Course : CPSC 5010

## Homework: 5

Problem 1:

Source code and Results:



The screenshot displays a C++ development environment with two main panes. The left pane shows the terminal output, and the right pane shows the source code for `problem1.cpp`.

**Terminal Output (Left Pane):**

```
> cd "/Users/student/Desktop/C++/Homeworks/HW5/" && g++ problem1.cpp -o problem1 && "/Users/student/Desktop/C++/Homeworks/HW5/"problem1
Enter first integer: 7
Enter second integer: 70
The gcd of 7 and 70 is :7
> cd "/Users/student/Desktop/C++/Homeworks/HW5/" && g++ problem1.cpp -o problem1 && "/Users/student/Desktop/C++/Homeworks/HW5/"problem1
Enter first integer: 50
Enter second integer: 6
The gcd of 50 and 6 is :2
```

**Source Code (Right Pane):**

```
Homeworks > HW5 > problem1.cpp > main()
1  #include<iostream>
2  using namespace std;
3  //Part 1: Define the prototype of the function
4  int gcd(int x, int y);
5
6  int main(){
7      //part 3: Prompt the user to input two integers
8      int x, y;
9      cout<< "Enter first integer: ";
10     cin >> x;
11     cout << endl;
12     cout<< "Enter second integer: ";
13     cin >> y;
14
15     cout<<"The gcd of "<<x << " and " << y<< " is :";
16     cout << gcd(x, y) << endl;
17
18     return 0;
19 }
20 //Part 2: implement the function to return the gcd
21 int gcd(int x, int y){
22     if(x % y == 0){
23         return y;
24     }
25     else {
26         return gcd(y, x%y);
27     }
28 }
29
```

Please click on the highlighted coding video link : [Problem 1](#)

Problem 2:  
Source code and Results:

The screenshot displays a C++ IDE with two main panels. The left panel shows the terminal output, and the right panel shows the source code for `problem8.cpp`.

**Terminal Output (Left Panel):**

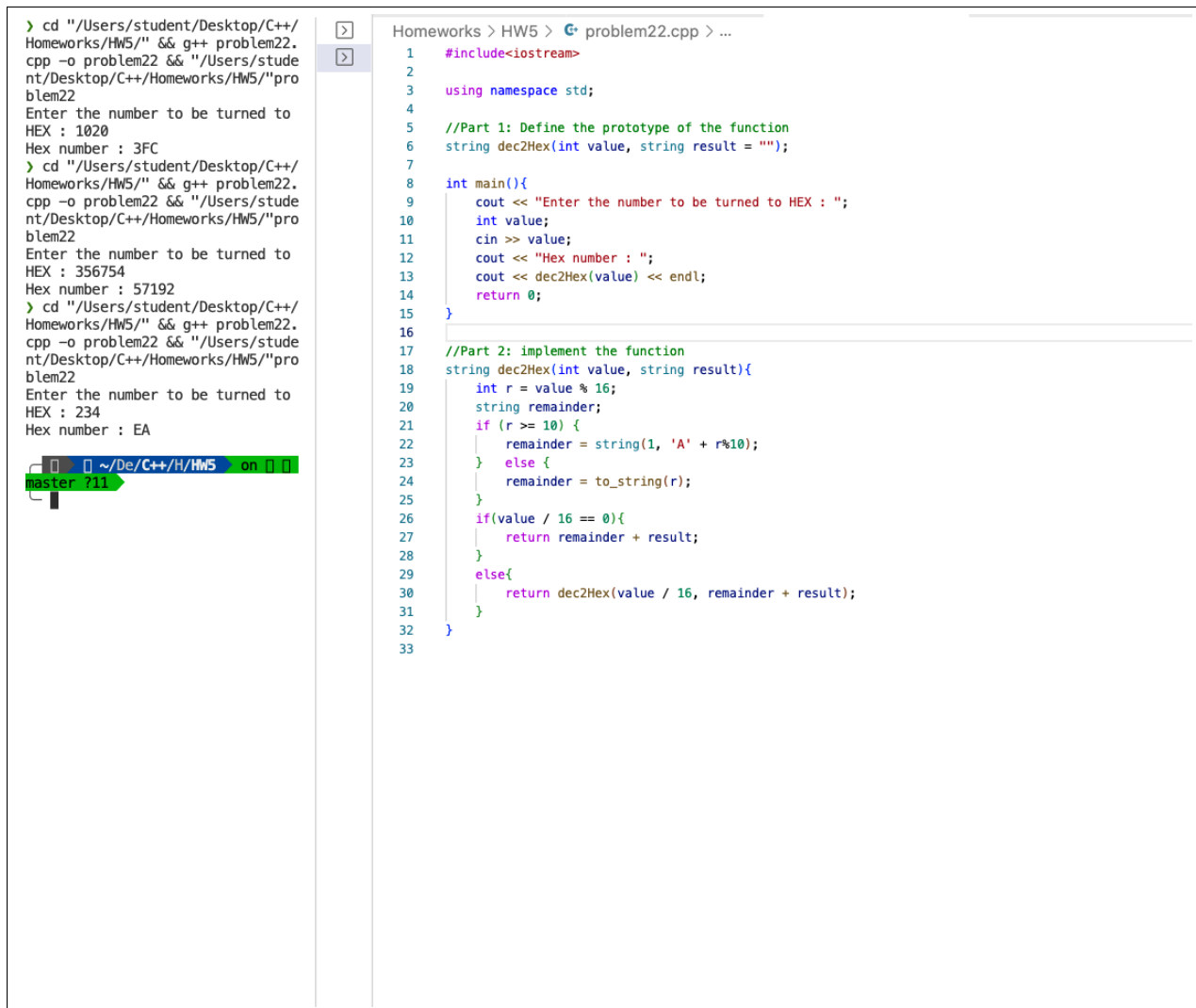
```
> cd "/Users/student/Desktop/C++/Homeworks/HW5/" && g++ problem8.cpp -o problem8 && "/Users/student/Desktop/C++/Homeworks/HW5/"problem8
Enter the number to be reversed : 123456789
Reversed number : 987654321
```

**Source Code (Right Panel):**

```
Homeworks > HW5 > problem8.cpp > main()
1  #include<iostream>
2
3  using namespace std;
4  //Part 1: Define the prototype of the function
5  void reverseDisplay(int number);
6
7  int main(){
8      cout << "Enter the number to be reversed : ";
9      int number;
10     cin >> number;
11     cout << "Reversed number : ";
12     reverseDisplay(number);
13     return 0;
14 }
15 //Part 2: implement the function
16 void reverseDisplay(int number){
17     if (number < 10){
18         cout << number;
19     }
20     else{
21         cout << number % 10;
22         reverseDisplay(number / 10); // calling the function recursively
23     }
24 }
```

Please click on the highlighted coding video link : [Problem 2](#)

### Problem 3: Source code and Results:



The screenshot displays a C++ IDE with two main panels. The left panel shows the terminal output of a program that converts decimal numbers to hexadecimal. The right panel shows the source code of the program, which includes a recursive function for conversion.

**Terminal Output (Left Panel):**

```
> cd "/Users/student/Desktop/C++/Homeworks/HW5/" && g++ problem22.cpp -o problem22 && "/Users/student/Desktop/C++/Homeworks/HW5/"problem22
Enter the number to be turned to HEX : 1020
Hex number : 3FC
> cd "/Users/student/Desktop/C++/Homeworks/HW5/" && g++ problem22.cpp -o problem22 && "/Users/student/Desktop/C++/Homeworks/HW5/"problem22
Enter the number to be turned to HEX : 356754
Hex number : 57192
> cd "/Users/student/Desktop/C++/Homeworks/HW5/" && g++ problem22.cpp -o problem22 && "/Users/student/Desktop/C++/Homeworks/HW5/"problem22
Enter the number to be turned to HEX : 234
Hex number : EA
```

**Source Code (Right Panel):**

```
Homeworks > HW5 > problem22.cpp > ...
1  #include<iostream>
2
3  using namespace std;
4
5  //Part 1: Define the prototype of the function
6  string dec2Hex(int value, string result = "");
7
8  int main(){
9      cout << "Enter the number to be turned to HEX : ";
10     int value;
11     cin >> value;
12     cout << "Hex number : ";
13     cout << dec2Hex(value) << endl;
14     return 0;
15 }
16
17 //Part 2: implement the function
18 string dec2Hex(int value, string result){
19     int r = value % 16;
20     string remainder;
21     if (r >= 10) {
22         remainder = string(1, 'A' + r%10);
23     } else {
24         remainder = to_string(r);
25     }
26     if(value / 16 == 0){
27         return remainder + result;
28     }
29     else{
30         return dec2Hex(value / 16, remainder + result);
31     }
32 }
33
```

Please click on the highlighted coding video link : [Problem 3](#)

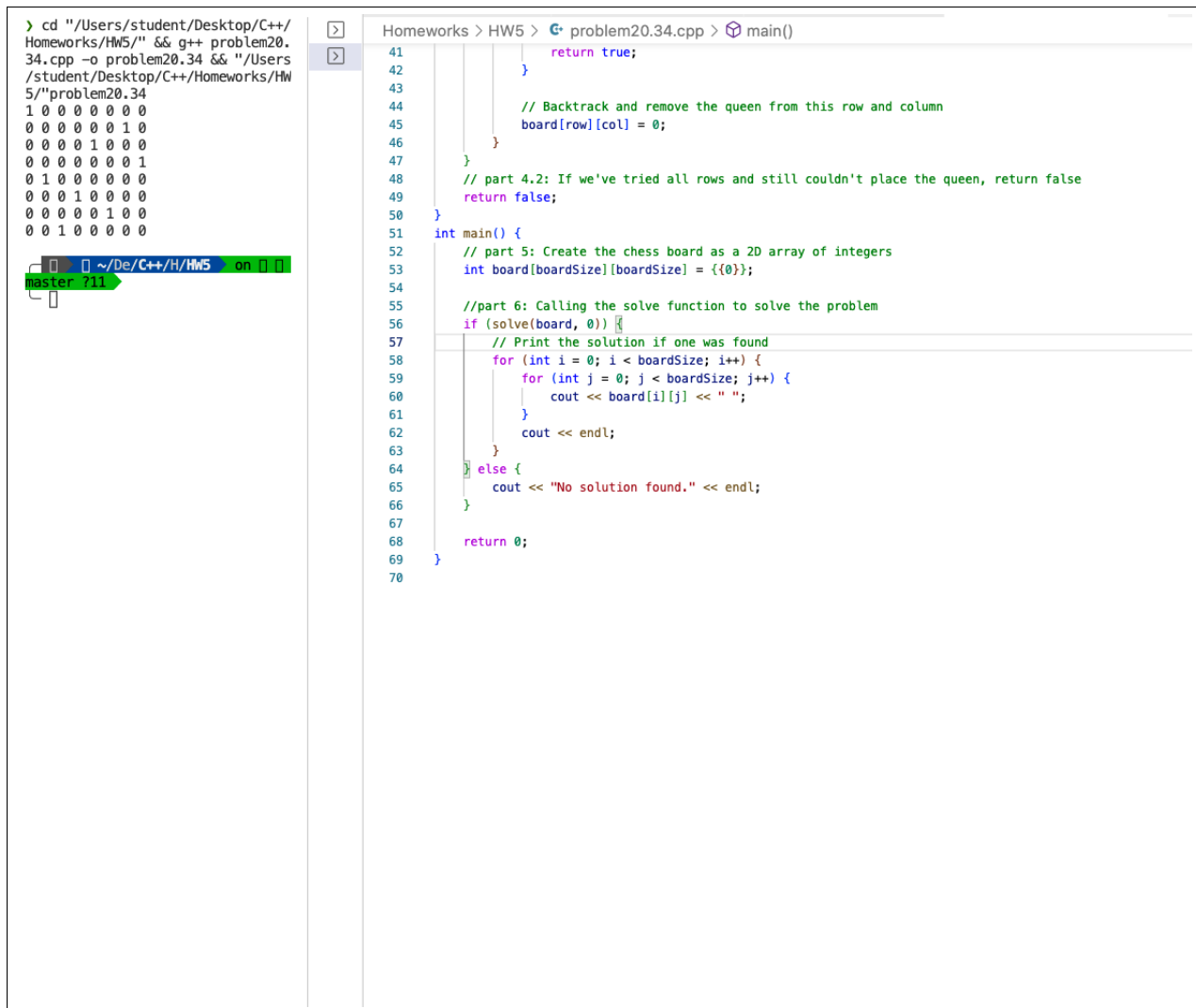
Problem 4.1 :  
Source code and Results:

```
> cd "/Users/student/Desktop/C++/Homeworks/HW5/" && g++ problem20.34.cpp -o problem20.34 && "/Users/student/Desktop/C++/Homeworks/HW5/"problem20.34
1 0 0 0 0 0 0 0
0 0 0 0 0 0 1 0
0 0 0 0 1 0 0 0
0 0 0 0 0 0 0 1
0 1 0 0 0 0 0 0
0 0 0 1 0 0 0 0
0 0 0 0 0 1 0 0
0 0 1 0 0 0 0 0

Homeworks > HW5 > problem20.34.cpp > queenIsSafe(int(*)[boardSize], int, int)
1  #include <iostream>
2  using namespace std;
3  const int boardSize = 8;
4  // Part 1: checking if a queen can be placed at a given row
5  bool queenIsSafe(int (*board)[boardSize], int row, int col){
6      for(int i = 0; i < col; i++){
7          if(board[row][i] == 1){
8              return false;
9          }
10     }
11     // Part 1.1: check the left upper diagonal
12     for(int i = row, j = col; i >= 0 && j >= 0; i--, j--){
13         if(board[i][j] == 1){
14             return false;
15         }
16     }
17     //part 1.2: check the left lower diagonal
18     for(int i = row, j = col; i < boardSize && j >= 0; i++, j--){
19         if(board[i][j] == 1){
20             return false;
21         }
22     }
23     // Part 1.3: check the queen can be placed safely
24     return true;
25 }
26 // Part 4: Recursive function to solve the == problem
27 bool solve(int (*board)[boardSize], int col) {
28     // Base case: all queens have been placed successfully
29     if (col >= boardSize) {
30         return true;
31     }
32     // part 4.1 : check by placing the queen in each row of the current column
33     for (int row = 0; row < boardSize; ++row) {
34         // part 4.2: Checking if it's safe to place the queen in this row and column
35         if (queenIsSafe(board, row, col)) {
36             // Place the queen in this row and column
37             board[row][col] = 1;
38
39             // Recursively solve the problem for the next column
40             if (solve(board, col + 1)) {
41                 return true;
42             }
43
44             // Backtrack and remove the queen from this row and column
45             board[row][col] = 0;
46         }
47     }
48     // part 4.2: If we've tried all rows and still couldn't place the queen, return false
49     return false;
50 }
51 int main() {
52     // part 5: Create the chess board as a 2D array of integers
53     int board[boardSize][boardSize] = {{0}};
```

Please click on the highlighted coding video link : [Problem 4.1](#)

Problem 4.2 :  
Source code and Results:



The screenshot displays a C++ IDE with two panels. The left panel shows the output of a program, which is an 8x8 chessboard represented by a grid of 0s and 1s. The right panel shows the source code for 'problem20.34.cpp'. The code implements a backtracking algorithm to solve the N-Queens problem. It includes a recursive function 'solve' that places queens row by row, checking for conflicts. The 'main' function initializes an 8x8 board and calls 'solve' to find a solution. The output in the left panel shows a valid solution where each row and column contains exactly one queen (represented by a 1).

```
> cd "/Users/student/Desktop/C++/Homeworks/HW5/" && g++ problem20.34.cpp -o problem20.34 && "/Users/student/Desktop/C++/Homeworks/HW5/problem20.34"
1 0 0 0 0 0 0 0
0 0 0 0 0 0 1 0
0 0 0 0 1 0 0 0
0 0 0 0 0 0 0 1
0 1 0 0 0 0 0 0
0 0 0 1 0 0 0 0
0 0 0 0 0 1 0 0
0 0 1 0 0 0 0 0

Homeworks > HW5 > problem20.34.cpp > main()
41         return true;
42     }
43
44     // Backtrack and remove the queen from this row and column
45     board[row][col] = 0;
46 }
47
48 // part 4.2: If we've tried all rows and still couldn't place the queen, return false
49 return false;
50 }
51
52 int main() {
53     // part 5: Create the chess board as a 2D array of integers
54     int board[boardSize][boardSize] = {{0}};
55
56     //part 6: Calling the solve function to solve the problem
57     if (solve(board, 0)) {
58         // Print the solution if one was found
59         for (int i = 0; i < boardSize; i++) {
60             for (int j = 0; j < boardSize; j++) {
61                 cout << board[i][j] << " ";
62             }
63             cout << endl;
64         }
65     } else {
66         cout << "No solution found." << endl;
67     }
68
69     return 0;
70 }
```

Please click on the highlighted coding video link : [Problem 4.2](#)