Assignment # 3

INSTRUCTIONS:

- Copied tasks will be awarded zero without any investigation.
- In case two or more students are found to have copied from each other, all involved will receive a score of zero. No arguments or disputes will be entertained.
- Assignment after the due date will not be accepted.
- Understanding questions is part of the assignment.
- Plagiarism of any shape or form will not be tolerated. In case of plagiarism, will receive a zero mark for the assignment
- Submit a separate .cpp file for each question.
- Please refrain from using advanced concepts in the assignment; it would lead to zero marks.
- Strictly follow this naming convention for .cpp file name_rollno_section_questionnumber.cpp failure to do so will result in zero marks.
- Ensure that your assignment is uploaded using your NU ID. Submissions made without using the NU ID will lead to a zero mark.

Problem 1:

You are tasked with writing a program that finds and prints a special sequence of integers in reverse order using while loop, following these rules:

Starting number: The user inputs a positive integer N.
Reduction process: Starting from N, repeatedly do the following:
If the number is even, divide it by 2.
If the number is odd, multiply it by 3 and add 1.
Continue until the number reaches 1.
Reversing the sequence: Print all the numbers encountered during this process in reverse order (i.e., starting from 1 and ending at N).
Loop Condition: You must use a **while loop** to implement the process. Do not use any other types of loops.
Challenge: You are not allowed to use arrays, vectors, or any data structures that would directly store the sequence beforehand. Instead, you must find a way to reverse the output through logic.

Sample Output:

```
Enter N: 6
Sequence in reverse order: 1 2 4 8 16 5 10 3 6
```

Understanding sample output:
If you start with N = 6, the sequence is generated as follows:

Start with 6.
6 is even → divide by 2 → result is 3.
3 is odd → multiply by 3 and add 1 → result is 10.
10 is even → divide by 2 → result is 5.
5 is odd → multiply by 3 and add 1 → result is 16.
16 is even → divide by 2 → result is 8.
8 is even → divide by 2 → result is 4.
4 is even → divide by 2 → result is 2.
2 is even → divide by 2 → result is 1.
So the sequence is: 6, 3, 10, 5, 16, 8, 4, 2, 1.

Problem 2:

Create a more sophisticated banking system simulation that allows a user to manage their account with additional features, while still enabling basic operations like deposit, withdrawal, and balance checking using do while loop.

Requirements:

Initial Setup:
Start with an initial balance of $5000.

Operations Menu:
Prompt the user to choose from the following options:
Deposit money
Withdraw money
Check balance
Transfer money to another account (implement this with a simple hardcoded recipient account)
Exit the program

Balance Management:
After each operation, display the updated balance.
Ensure that the withdrawal amount does not exceed the current balance.
Validate the deposit amount to ensure it is positive.

Transfer Feature:
Allow the user to transfer a specified amount to a hardcoded recipient account with a fixed balance (e.g., $500).
Ensure that the user has enough balance to complete the transfer.
Update both the user's balance and the recipient account balance accordingly.

Looping and User Prompts:
After each operation, ask the user if they want to perform another action.
Use a do-while loop to allow for repeated operations within the main menu.
Use another do-while loop for confirming the action after each operation.

Exit Confirmation:
When the user chooses to exit, ask for confirmation to ensure they want to terminate the program.

Sample Output:

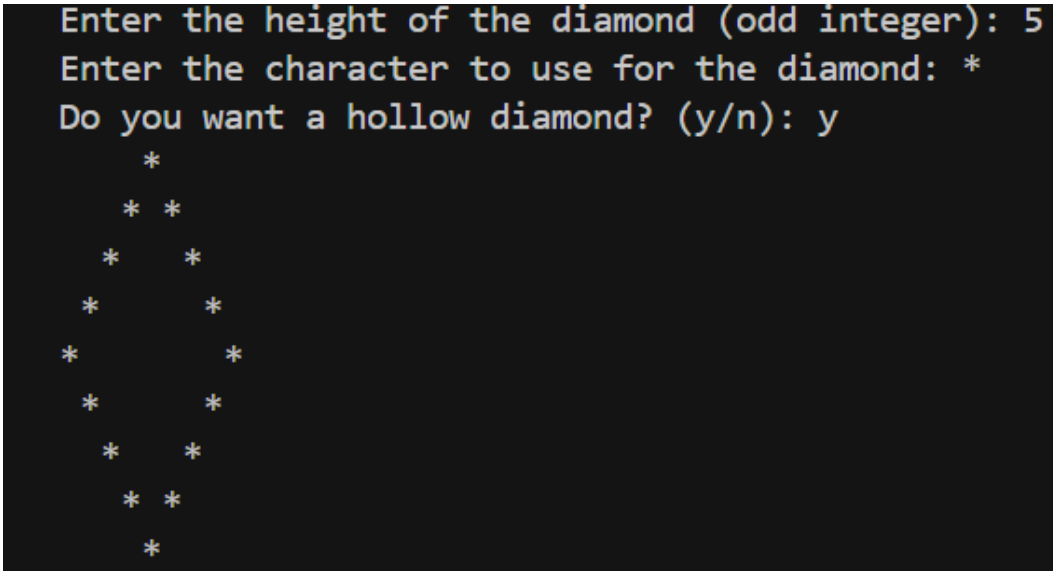
```
Welcome to the FAST Banking System!
Choose an option:
1. Deposit money
2. Withdraw money
3. Check balance
4. Transfer money to another account
5. Exit
1
Enter amount to deposit: 100
Deposited: $100
New Balance: $5100
Do you want to perform another action? (y/n): y

Choose an option:
1. Deposit money
2. Withdraw money
3. Check balance
4. Transfer money to another account
5. Exit
2
Enter amount to withdraw: 100
Withdrew: $100
New Balance: $5000
Do you want to perform another action? (y/n): y

Choose an option:
1. Deposit money
2. Withdraw money
3. Check balance
4. Transfer money to another account
5. Exit
3
Current Balance: $5000
Do you want to perform another action? (y/n): y

Choose an option:
1. Deposit money
2. Withdraw money
3. Check balance
4. Transfer money to another account
5. Exit
4
Enter amount to transfer to recipient: 100
Transferred: $100 to recipient
Your New Balance: $4900
Recipient's New Balance: $600
```

Problem 3:

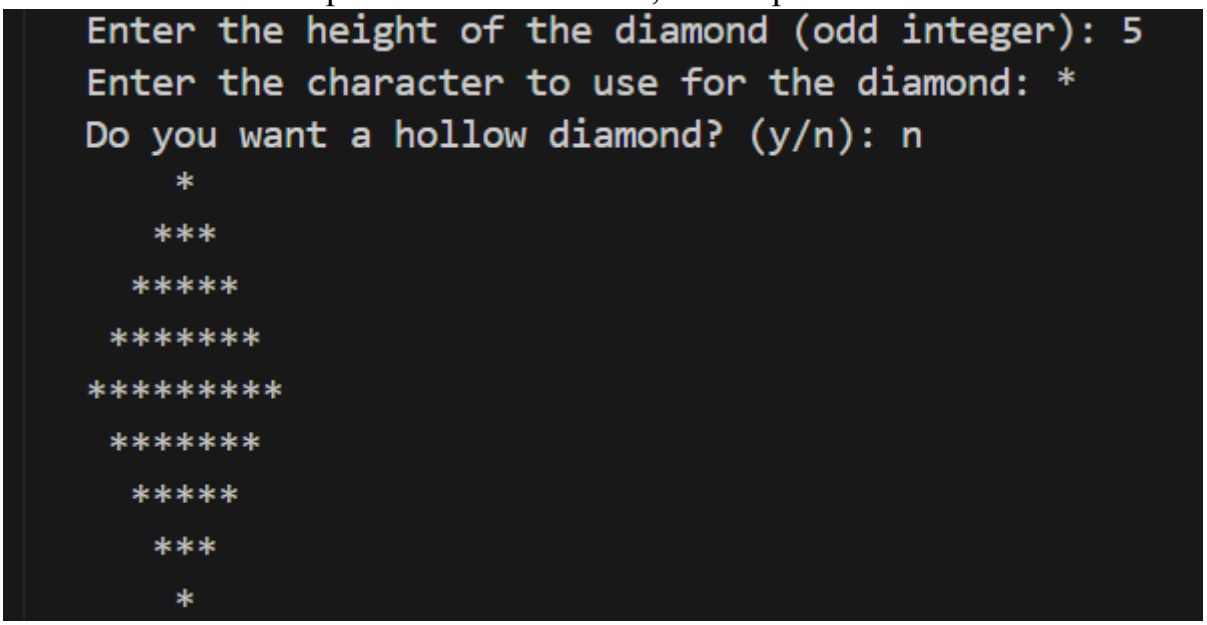
Write a C++ program that prints a customizable diamond pattern based on user input using for, nested for loop.

Requirements:
Input:
Prompt the user to enter the height of the diamond, which must be an odd integer (e.g., 5, 7, 9).
Prompt the user to enter a character that will be used to draw the diamond (e.g., *, #, etc.).
Ask the user if they want to print a hollow diamond or a filled diamond by entering y for hollow or n for filled.

Sample Output:
If the user chooses to print a hollow diamond, the output should look as follows for a height of 5 and the character *:



If the user chooses to print a filled diamond, the output should look like this:



Problem 4:

Write a program that generates a multiplication table for a user-defined range using for and nested for loop. The program should prompt the user for two numbers: the starting and ending values for the multiplication table. Then, it should print the multiplication table for all numbers in that range, including:

Example Output

If the user inputs 3 as the starting value and 5 as the ending value, the output should look like this:

```
Enter the starting number: 3
Enter the ending number: 5
Multiplication Table from 3 to 5:
        3       4       5
3       9       12      15      | 36
4       12      16      20      | 48
5       15      20      25      | 60
-----------------------
Total: 144
```

Row Sums:

After printing each row of the multiplication results, you will calculate and display the sum of the products in that row.

Total Sum:

You will compute the total sum of all the products across all rows and display that total at the end of the table.

Understanding sample output:

For 3:

3 * 3 = 9

3 * 4 = 12

3 * 5 = 15

For 4:

4 * 3 = 12

4 * 4 = 16

4 * 5 = 20

For 5:

5 * 3 = 15

5 * 4 = 20

5 * 5 = 25