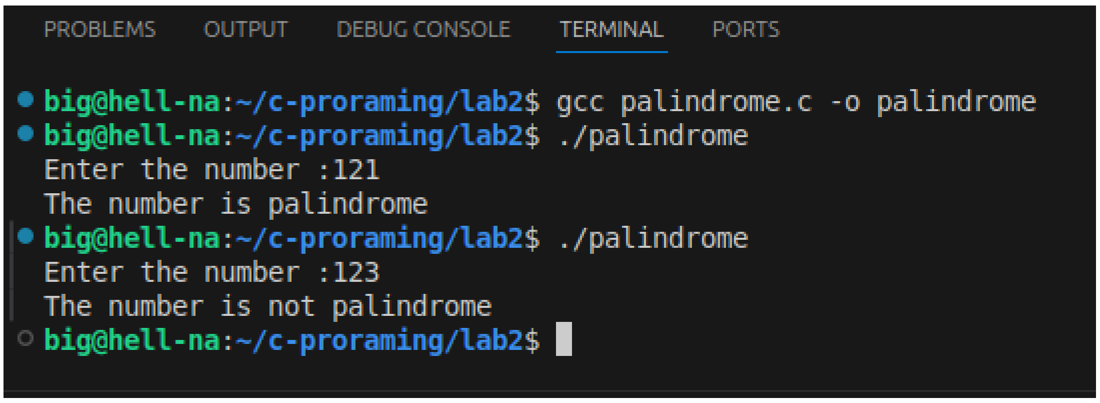
1. **Write a C program to take any number as user input and check whether it is palindrome or not.**

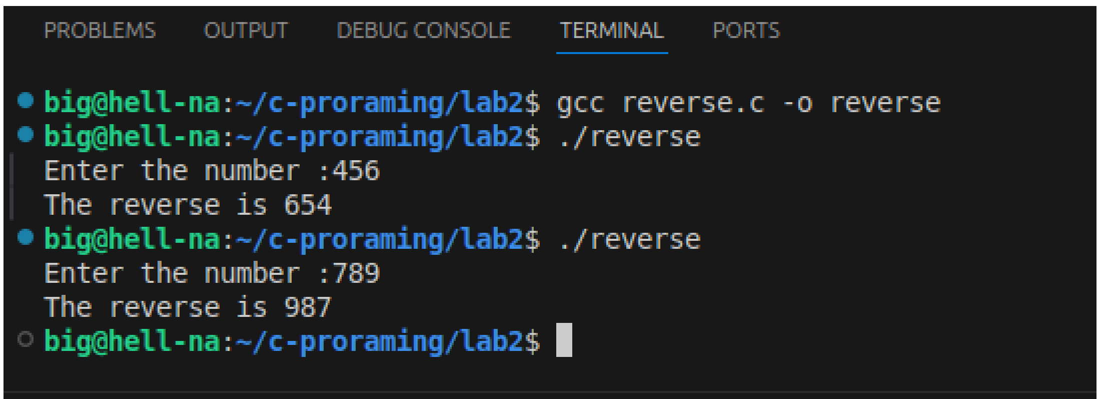
Output:



```
big@hell-na:~/c-proraming/lab2$ gcc palindrome.c -o palindrome
big@hell-na:~/c-proraming/lab2$ ./palindrome
Enter the number :121
The number is palindrome
big@hell-na:~/c-proraming/lab2$ ./palindrome
Enter the number :123
The number is not palindrome
big@hell-na:~/c-proraming/lab2$
```

2. **Write a C program to find a reverse of a number input by the user.**

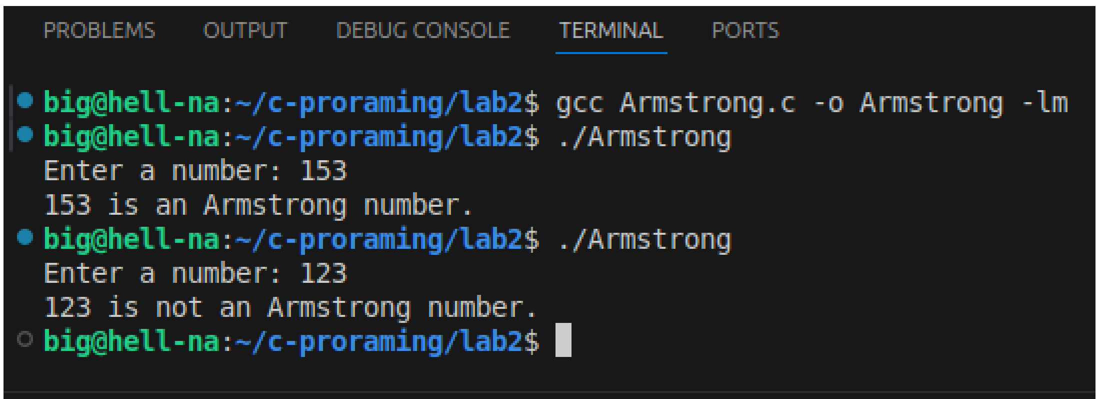
Output:



```
big@hell-na:~/c-proraming/lab2$ gcc reverse.c -o reverse
big@hell-na:~/c-proraming/lab2$ ./reverse
Enter the number :456
The reverse is 654
big@hell-na:~/c-proraming/lab2$ ./reverse
Enter the number :789
The reverse is 987
big@hell-na:~/c-proraming/lab2$
```

3. **Write a C program to take any number as user input and check whether it is Armstrong number or not.**

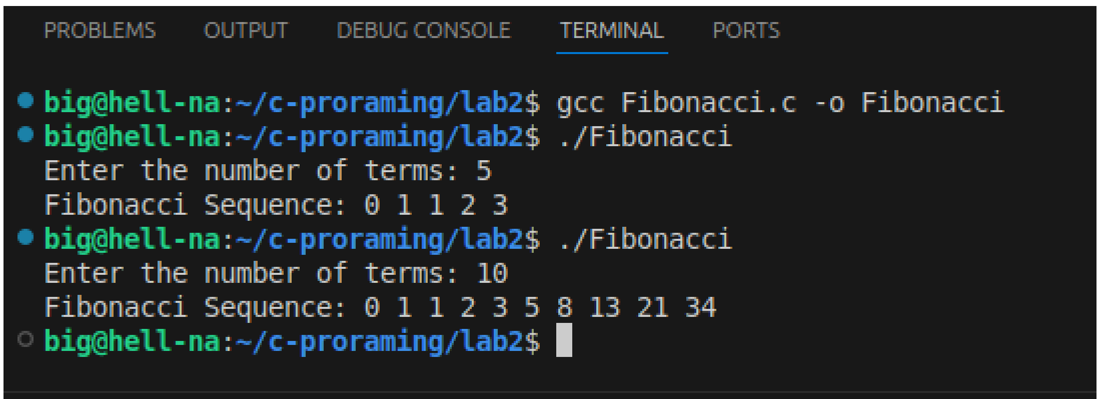
Output:



```
big@hell-na:~/c-proraming/lab2$ gcc Armstrong.c -o Armstrong -lm
big@hell-na:~/c-proraming/lab2$ ./Armstrong
Enter a number: 153
153 is an Armstrong number.
big@hell-na:~/c-proraming/lab2$ ./Armstrong
Enter a number: 123
123 is not an Armstrong number.
big@hell-na:~/c-proraming/lab2$ 
```

4. **Write a C program to generate Fibonacci sequence up-to n terms taking n as user input.**

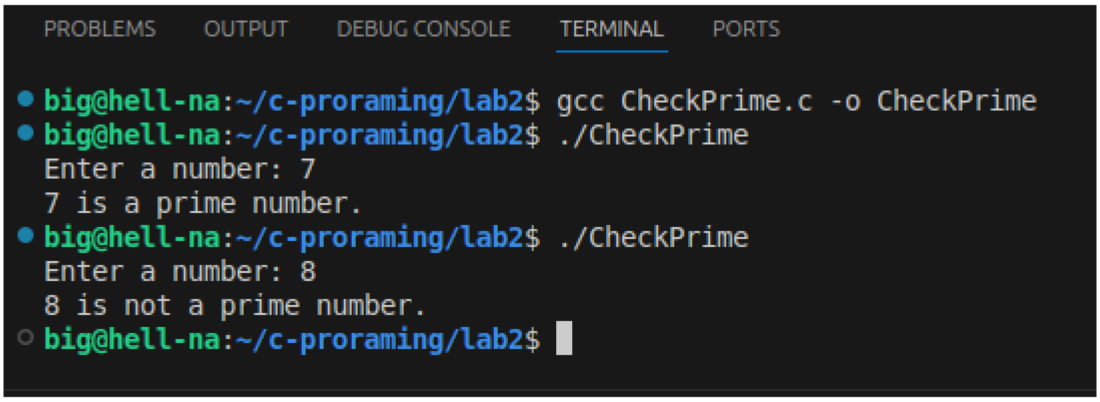
Output:



```
big@hell-na:~/c-proraming/lab2$ gcc Fibonacci.c -o Fibonacci
big@hell-na:~/c-proraming/lab2$ ./Fibonacci
Enter the number of terms: 5
Fibonacci Sequence: 0 1 1 2 3
big@hell-na:~/c-proraming/lab2$ ./Fibonacci
Enter the number of terms: 10
Fibonacci Sequence: 0 1 1 2 3 5 8 13 21 34
big@hell-na:~/c-proraming/lab2$
```

5. **Write a C program to check whether a number is prime or not.**

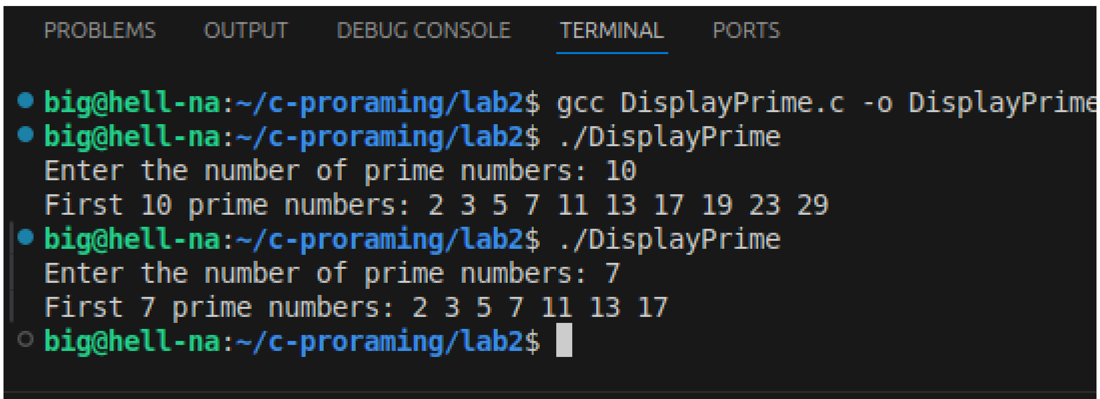
Output:



```
big@hell-na:~/c-proraming/lab2$ gcc CheckPrime.c -o CheckPrime
big@hell-na:~/c-proraming/lab2$ ./CheckPrime
Enter a number: 7
7 is a prime number.
big@hell-na:~/c-proraming/lab2$ ./CheckPrime
Enter a number: 8
8 is not a prime number.
big@hell-na:~/c-proraming/lab2$ 
```

6. **Write a C program to to display all the prime numbers up-to n terms.**

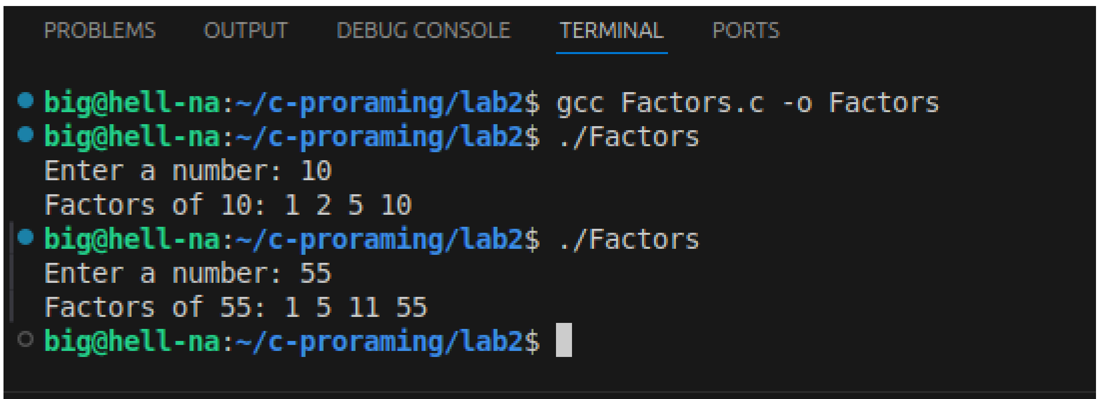
Output:



```
big@hell-na:~/c-proraming/lab2$ gcc DisplayPrime.c -o DisplayPrime
big@hell-na:~/c-proraming/lab2$ ./DisplayPrime
Enter the number of prime numbers: 10
First 10 prime numbers: 2 3 5 7 11 13 17 19 23 29
big@hell-na:~/c-proraming/lab2$ ./DisplayPrime
Enter the number of prime numbers: 7
First 7 prime numbers: 2 3 5 7 11 13 17
big@hell-na:~/c-proraming/lab2$
```

7. **Write a C program to to find all the factors of a natural number.**

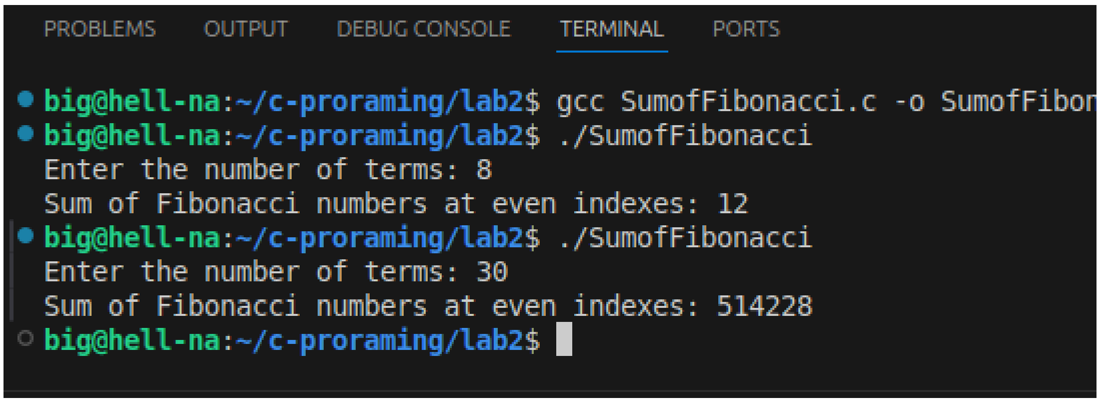
Output:



```
big@hell-na:~/c-proraming/lab2$ gcc Factors.c -o Factors
big@hell-na:~/c-proraming/lab2$ ./Factors
Enter a number: 10
Factors of 10: 1 2 5 10
big@hell-na:~/c-proraming/lab2$ ./Factors
Enter a number: 55
Factors of 55: 1 5 11 55
big@hell-na:~/c-proraming/lab2$
```

8. Write a C program to find the sum of Fibonacci numbers at Even indexes up to N terms.

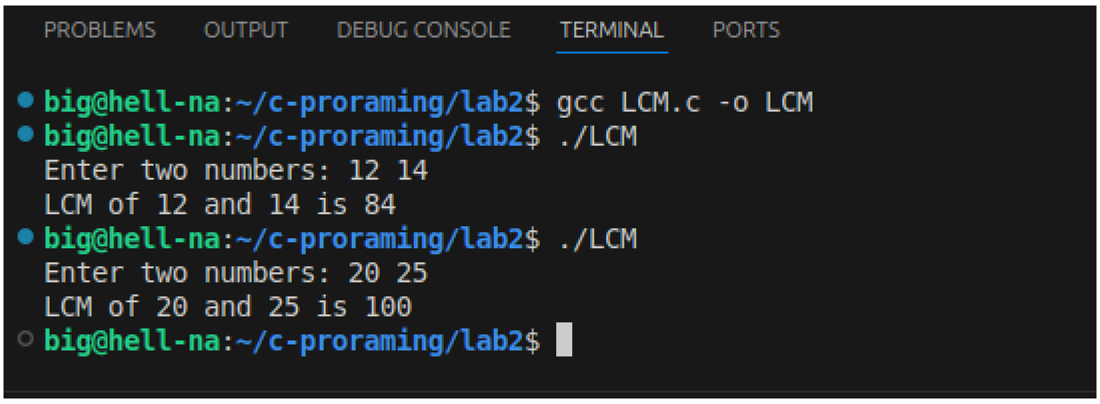
Output:



```
big@hell-na:~/c-proraming/lab2$ gcc SumofFibonacci.c -o SumofFibon
big@hell-na:~/c-proraming/lab2$ ./SumofFibonacci
Enter the number of terms: 8
Sum of Fibonacci numbers at even indexes: 12
big@hell-na:~/c-proraming/lab2$ ./SumofFibonacci
Enter the number of terms: 30
Sum of Fibonacci numbers at even indexes: 514228
big@hell-na:~/c-proraming/lab2$
```

9. **Write a C program to to find the LCM of two numbers.**

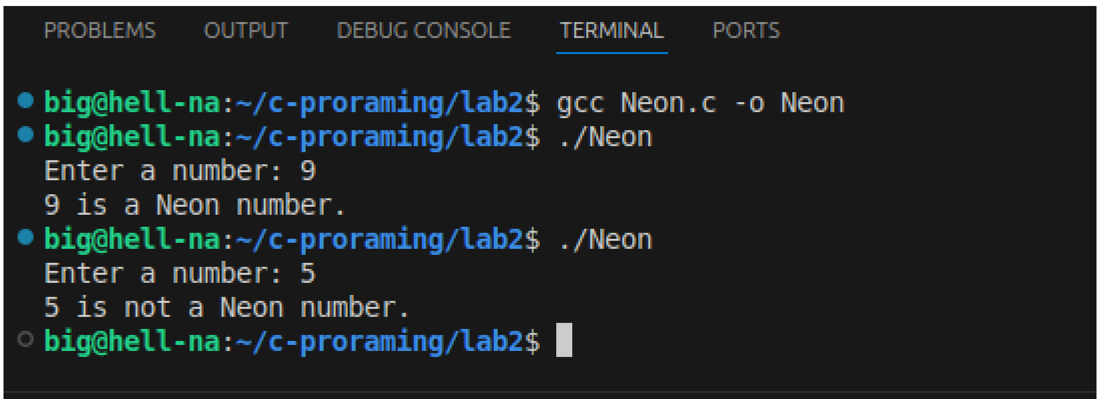
Output:



```
● big@hell-na:~/c-proraming/lab2$ gcc LCM.c -o LCM
● big@hell-na:~/c-proraming/lab2$ ./LCM
  Enter two numbers: 12 14
  LCM of 12 and 14 is 84
● big@hell-na:~/c-proraming/lab2$ ./LCM
  Enter two numbers: 20 25
  LCM of 20 and 25 is 100
○ big@hell-na:~/c-proraming/lab2$ 
```

10.Write a C program to check whether the number is neon number.

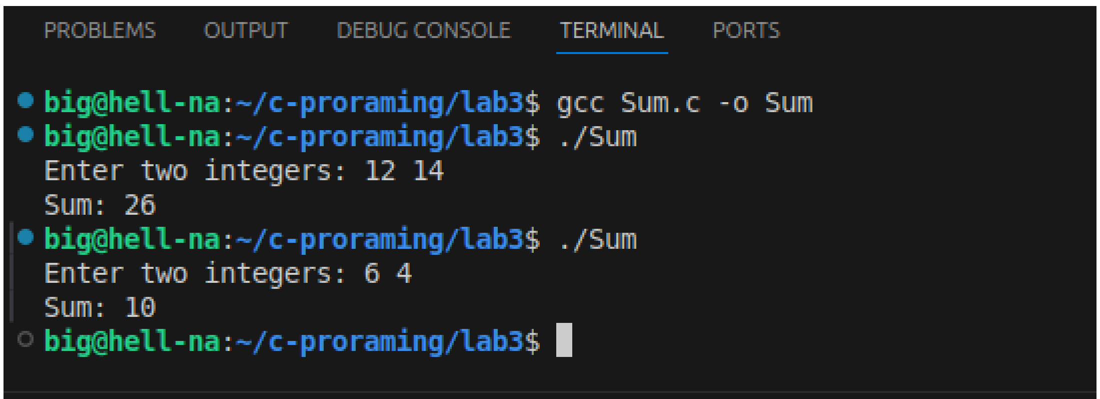
Output:



```
big@hell-na:~/c-proraming/lab2$ gcc Neon.c -o Neon
big@hell-na:~/c-proraming/lab2$ ./Neon
Enter a number: 9
9 is a Neon number.
big@hell-na:~/c-proraming/lab2$ ./Neon
Enter a number: 5
5 is not a Neon number.
big@hell-na:~/c-proraming/lab2$
```

1.Write a program to take two integers as input from the user and display their sum.

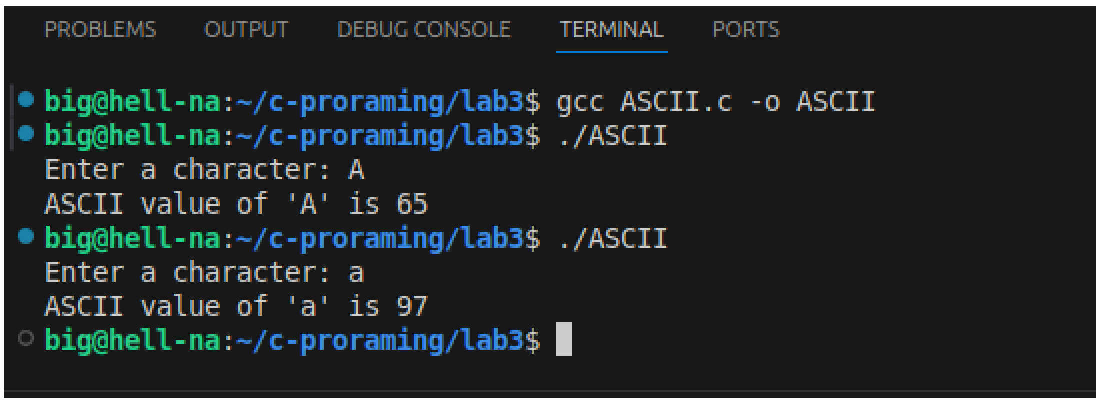
Output:



```
big@hell-na:~/c-proraming/lab3$ gcc Sum.c -o Sum
big@hell-na:~/c-proraming/lab3$ ./Sum
Enter two integers: 12 14
Sum: 26
big@hell-na:~/c-proraming/lab3$ ./Sum
Enter two integers: 6 4
Sum: 10
big@hell-na:~/c-proraming/lab3$
```

2. Write a program to take a character as input from the user and display its ASCII value.

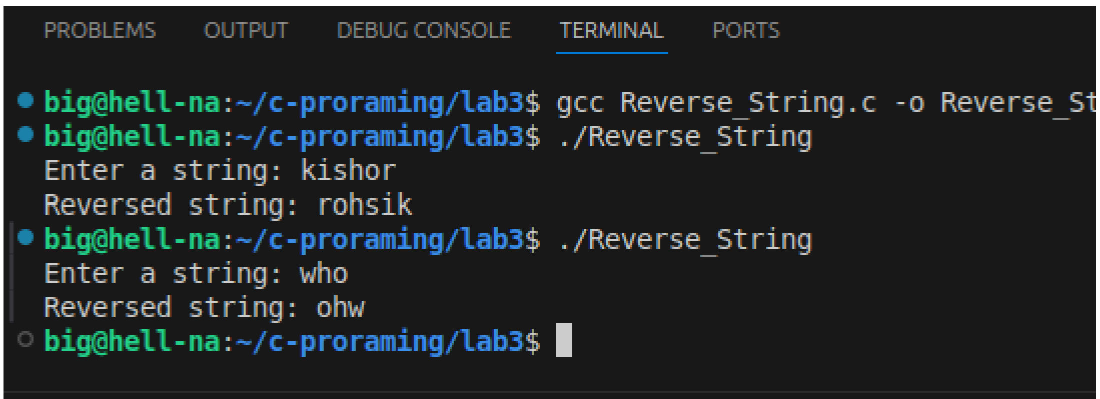
Output:



```
big@hell-na:~/c-proraming/lab3$ gcc ASCII.c -o ASCII
big@hell-na:~/c-proraming/lab3$ ./ASCII
Enter a character: A
ASCII value of 'A' is 65
big@hell-na:~/c-proraming/lab3$ ./ASCII
Enter a character: a
ASCII value of 'a' is 97
big@hell-na:~/c-proraming/lab3$
```

3. Write a program to take a string as input from the user and display it in reverse order.

Output:

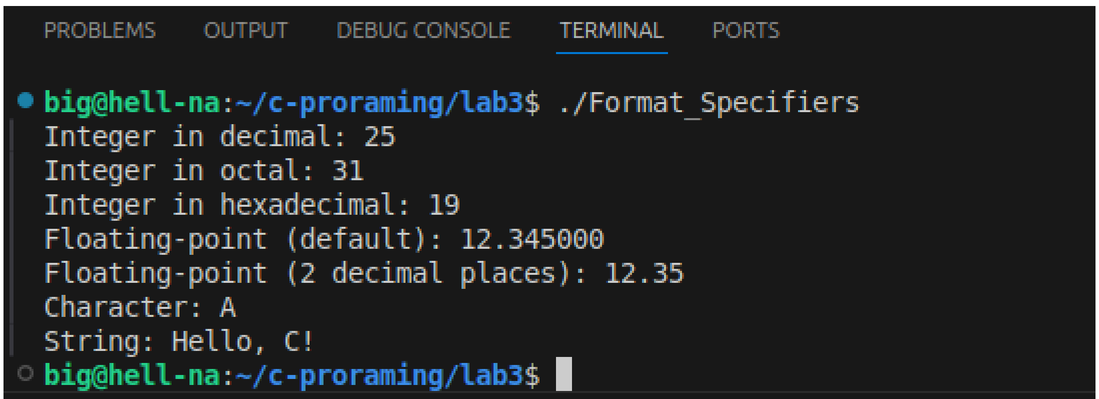


```
big@hell-na:~/c-proraming/lab3$ gcc Reverse_String.c -o Reverse_St
big@hell-na:~/c-proraming/lab3$ ./Reverse_String
Enter a string: kishor
Reversed string: rohsik
big@hell-na:~/c-proraming/lab3$ ./Reverse_String
Enter a string: who
Reversed string: ohw
big@hell-na:~/c-proraming/lab3$
```

4. Write a program that demonstrates the use of different format specifiers in the `printf`

function to display various types of data.
Your program should:Print an integer in decimal, octal, and hexadecimal formats.
Print a floating-point number with and without decimal places.
Print a character.Print a string

Output:



```
big@hell-na:~/c-proraming/lab3$ ./Format_Specifiers
Integer in decimal: 25
Integer in octal: 31
Integer in hexadecimal: 19
Floating-point (default): 12.345000
Floating-point (2 decimal places): 12.35
Character: A
String: Hello, C!
big@hell-na:~/c-proraming/lab3$
```