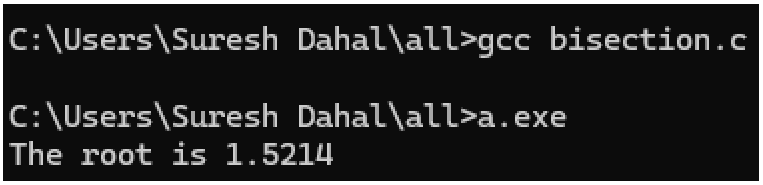
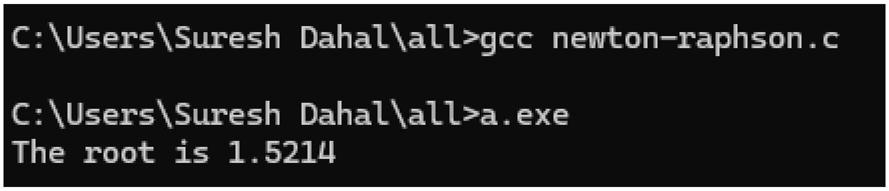
1. **Bisection Method**



```
C:\Users\Suresh Dahal\all>gcc bisection.c

C:\Users\Suresh Dahal\all>a.exe
The root is 1.5214
```

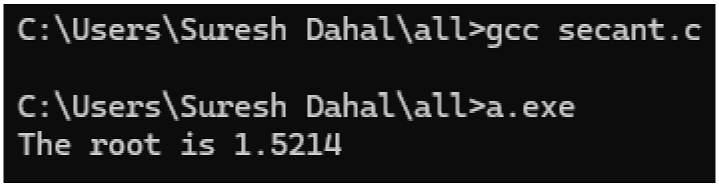
2. **Newton Raphson Method**



```
C:\Users\Suresh Dahal\all>gcc newton-raphson.c

C:\Users\Suresh Dahal\all>a.exe
The root is 1.5214
```

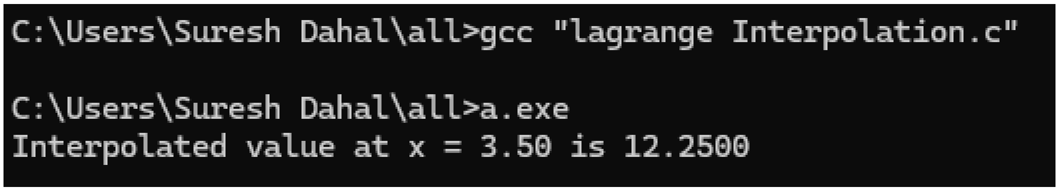
3. **Secant Method**



```
C:\Users\Suresh Dahal\all>gcc secant.c

C:\Users\Suresh Dahal\all>a.exe
The root is 1.5214
```

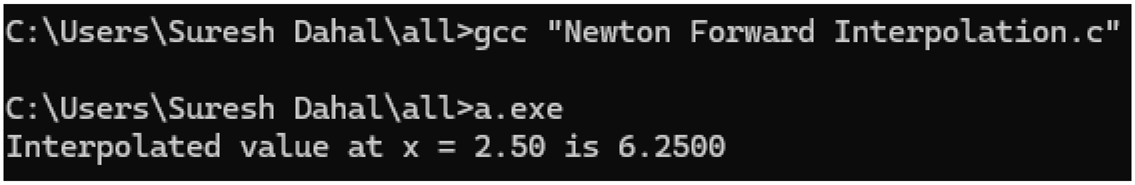
4. **Lagrange Interpolation**



```
C:\Users\Suresh Dahal\all>gcc "lagrange Interpolation.c"

C:\Users\Suresh Dahal\all>a.exe
Interpolated value at x = 3.50 is 12.2500
```

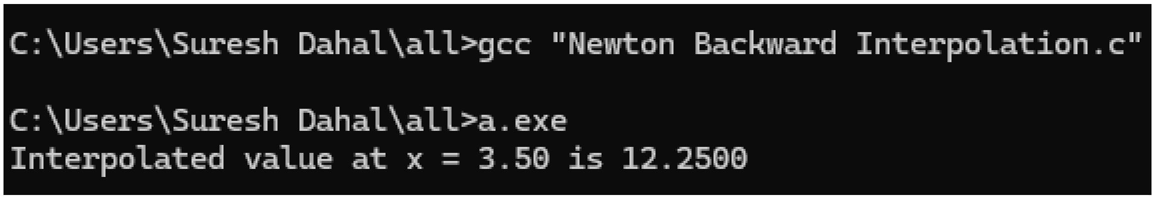
5. **Newton Forward Interpolation**



```
C:\Users\Suresh Dahal\all>gcc "Newton Forward Interpolation.c"

C:\Users\Suresh Dahal\all>a.exe
Interpolated value at x = 2.50 is 6.2500
```

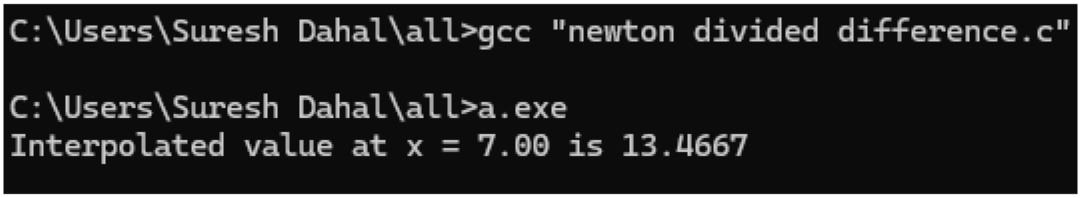
6. **Newton Backward Interpolation**



```
C:\Users\Suresh Dahal\all>gcc "Newton Backward Interpolation.c"

C:\Users\Suresh Dahal\all>a.exe
Interpolated value at x = 3.50 is 12.2500
```

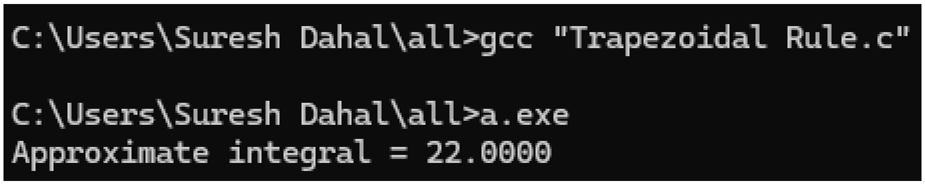
7. **Newton Divided Difference**



```
C:\Users\Suresh Dahal\all>gcc "newton divided difference.c"

C:\Users\Suresh Dahal\all>a.exe
Interpolated value at x = 7.00 is 13.4667
```

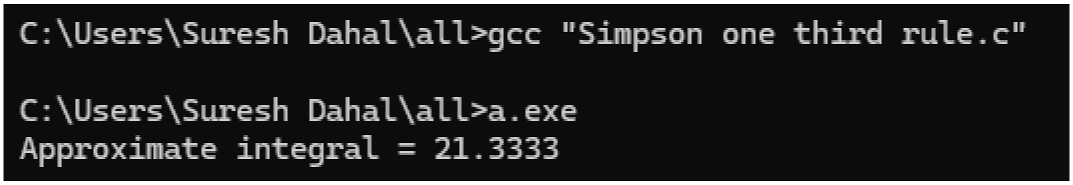
8. **Trapezoidal Rule**



```
C:\Users\Suresh Dahal\all>gcc "Trapezoidal Rule.c"

C:\Users\Suresh Dahal\all>a.exe
Approximate integral = 22.0000
```

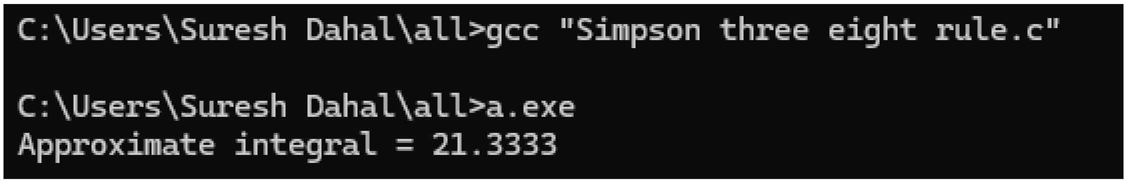
9. **Simpson’s 1/3 Rule**



```
C:\Users\Suresh Dahal\all>gcc "Simpson one third rule.c"

C:\Users\Suresh Dahal\all>a.exe
Approximate integral = 21.3333
```

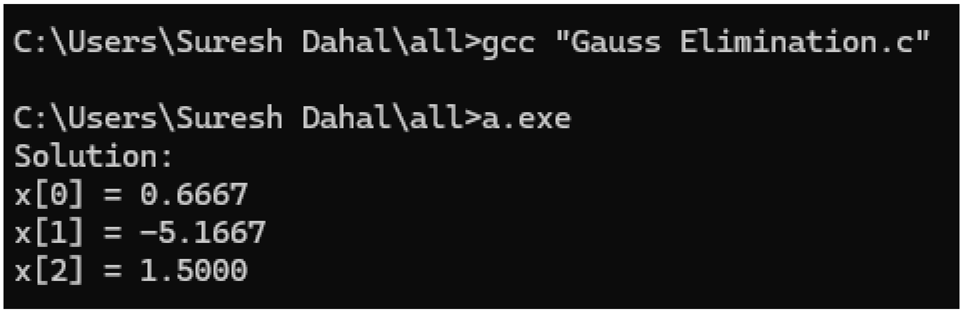
10. **Simpson’s 3/8 Rule**



```
C:\Users\Suresh Dahal\all>gcc "Simpson three eight rule.c"

C:\Users\Suresh Dahal\all>a.exe
Approximate integral = 21.3333
```

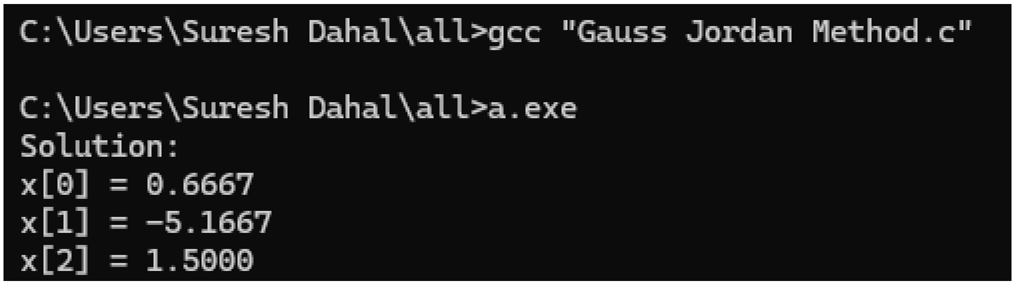
11. **Gauss Elimination Method**



```
C:\Users\Suresh Dahal\all>gcc "Gauss Elimination.c"

C:\Users\Suresh Dahal\all>a.exe
Solution:
x[0] = 0.6667
x[1] = -5.1667
x[2] = 1.5000
```

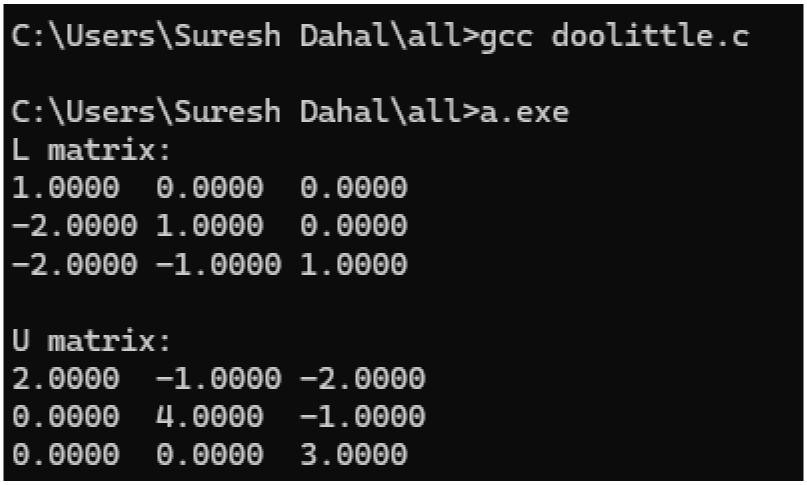
12. **Gauss-Jordan Method**



```
C:\Users\Suresh Dahal\all>gcc "Gauss Jordan Method.c"

C:\Users\Suresh Dahal\all>a.exe
Solution:
x[0] = 0.6667
x[1] = -5.1667
x[2] = 1.5000
```

13. **Matrix Factorization using Doolittle LU Decomposition**

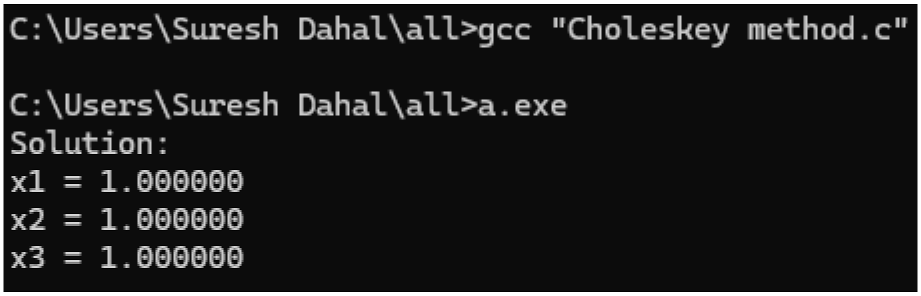


```
C:\Users\Suresh Dahal\all>gcc doolittle.c

C:\Users\Suresh Dahal\all>a.exe
L matrix:
1.0000  0.0000  0.0000
-2.0000 1.0000  0.0000
-2.0000 -1.0000 1.0000

U matrix:
2.0000  -1.0000 -2.0000
0.0000  4.0000  -1.0000
0.0000  0.0000  3.0000
```

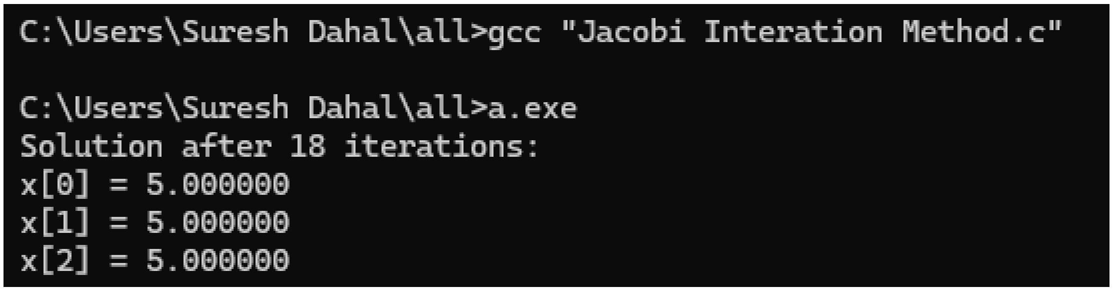
14. **Matrix Factorization using Cholesky's Method**



```
C:\Users\Suresh Dahal\all>gcc "Choleskey method.c"

C:\Users\Suresh Dahal\all>a.exe
Solution:
x1 = 1.000000
x2 = 1.000000
x3 = 1.000000
```

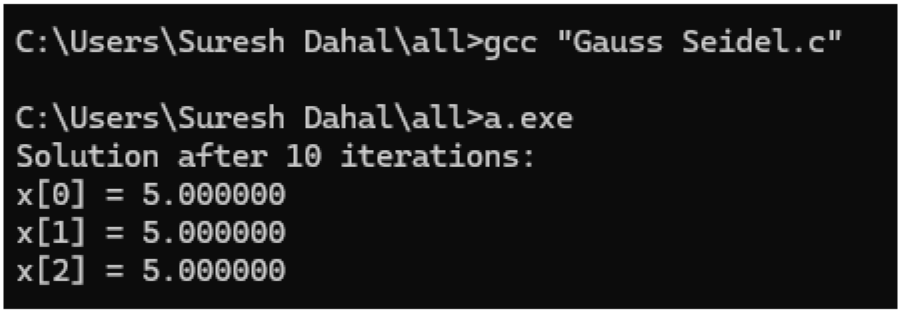
15. **Jacobi Iterative Method**



```
C:\Users\Suresh Dahal\all>gcc "Jacobi Interation Method.c"

C:\Users\Suresh Dahal\all>a.exe
Solution after 18 iterations:
x[0] = 5.000000
x[1] = 5.000000
x[2] = 5.000000
```

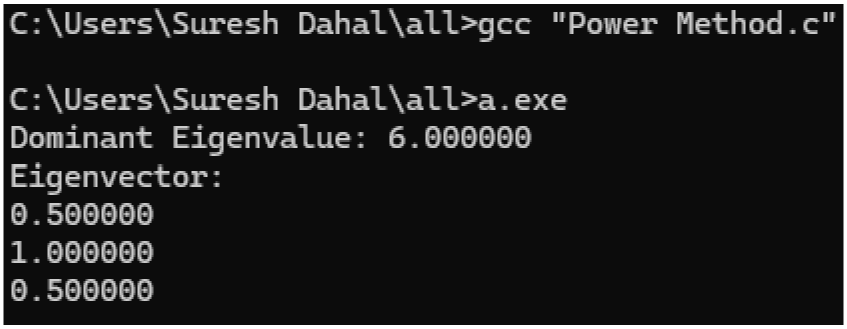
16. **Gauss-Seidel Iterative Method**



```
C:\Users\Suresh Dahal\all>gcc "Gauss Seidel.c"

C:\Users\Suresh Dahal\all>a.exe
Solution after 10 iterations:
x[0] = 5.000000
x[1] = 5.000000
x[2] = 5.000000
```

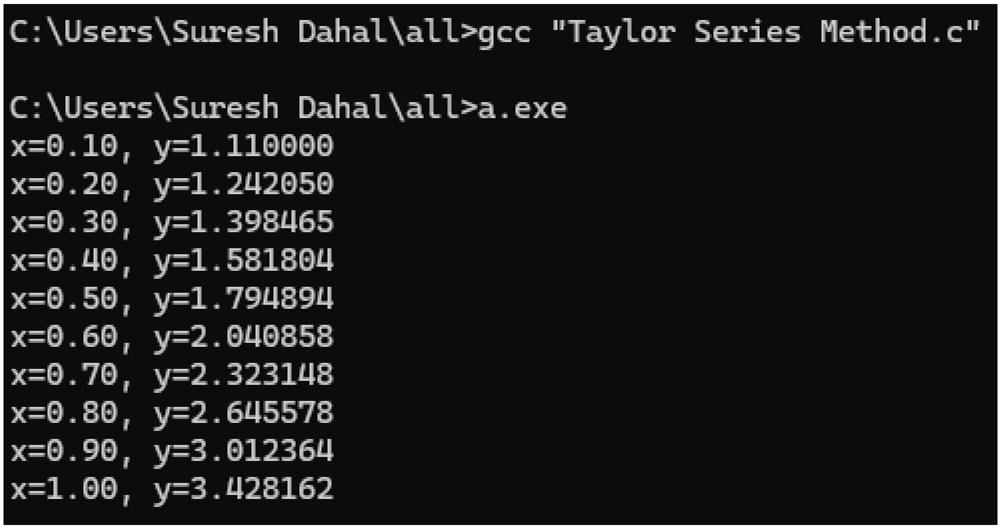
17. **Power Method**



```
C:\Users\Suresh Dahal\all>gcc "Power Method.c"

C:\Users\Suresh Dahal\all>a.exe
Dominant Eigenvalue: 6.000000
Eigenvector:
0.500000
1.000000
0.500000
```

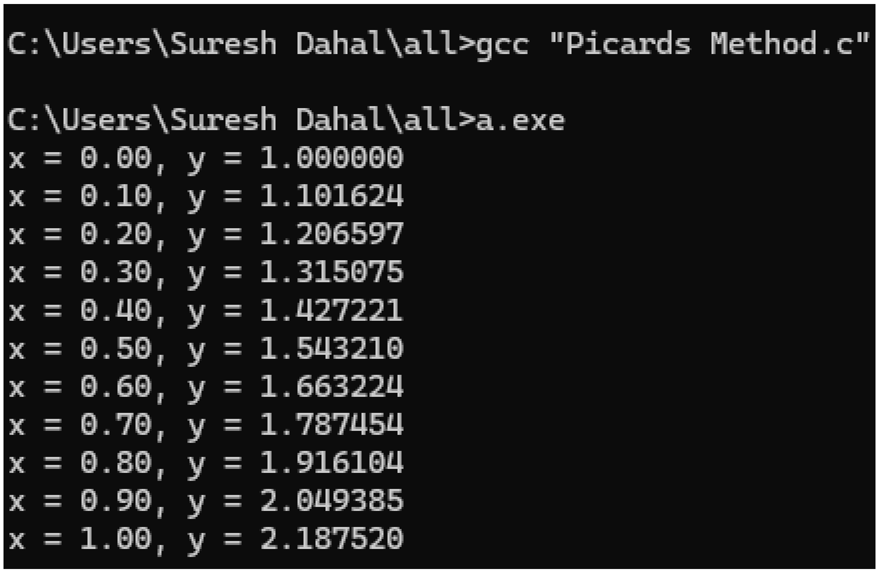
18. **Taylor Series Method for ODE**



```
C:\Users\Suresh Dahal\all>gcc "Taylor Series Method.c"

C:\Users\Suresh Dahal\all>a.exe
x=0.10, y=1.110000
x=0.20, y=1.242050
x=0.30, y=1.398465
x=0.40, y=1.581804
x=0.50, y=1.794894
x=0.60, y=2.040858
x=0.70, y=2.323148
x=0.80, y=2.645578
x=0.90, y=3.012364
x=1.00, y=3.428162
```

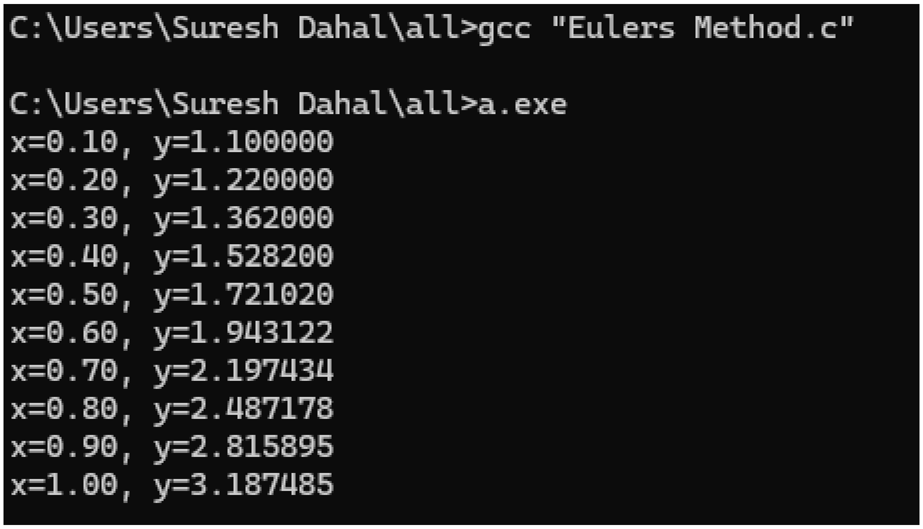
19. **Picard’s Method**



```
C:\Users\Suresh Dahal\all>gcc "Picards Method.c"

C:\Users\Suresh Dahal\all>a.exe
x = 0.00, y = 1.000000
x = 0.10, y = 1.101624
x = 0.20, y = 1.206597
x = 0.30, y = 1.315075
x = 0.40, y = 1.427221
x = 0.50, y = 1.543210
x = 0.60, y = 1.663224
x = 0.70, y = 1.787454
x = 0.80, y = 1.916104
x = 0.90, y = 2.049385
x = 1.00, y = 2.187520
```

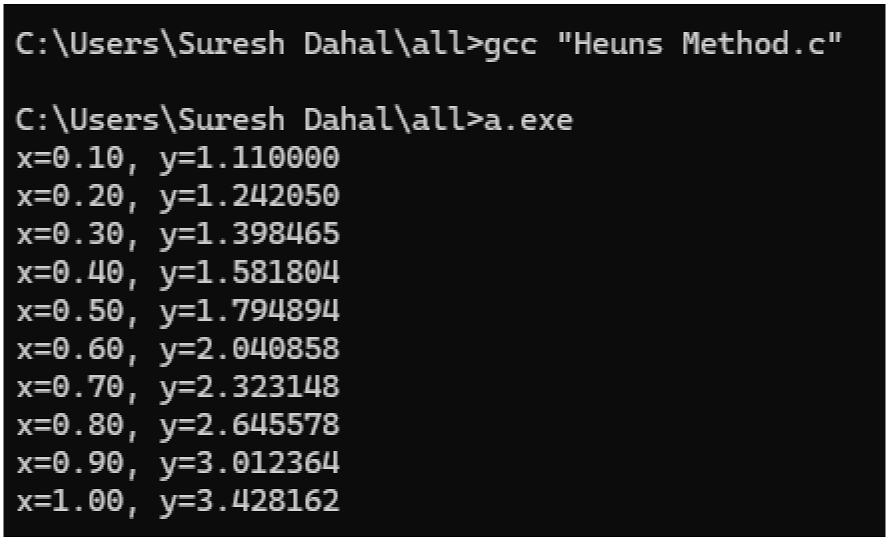
20. Euler’s Method



```
C:\Users\Suresh Dahal\all>gcc "Eulers Method.c"

C:\Users\Suresh Dahal\all>a.exe
x=0.10, y=1.100000
x=0.20, y=1.220000
x=0.30, y=1.362000
x=0.40, y=1.528200
x=0.50, y=1.721020
x=0.60, y=1.943122
x=0.70, y=2.197434
x=0.80, y=2.487178
x=0.90, y=2.815895
x=1.00, y=3.187485
```

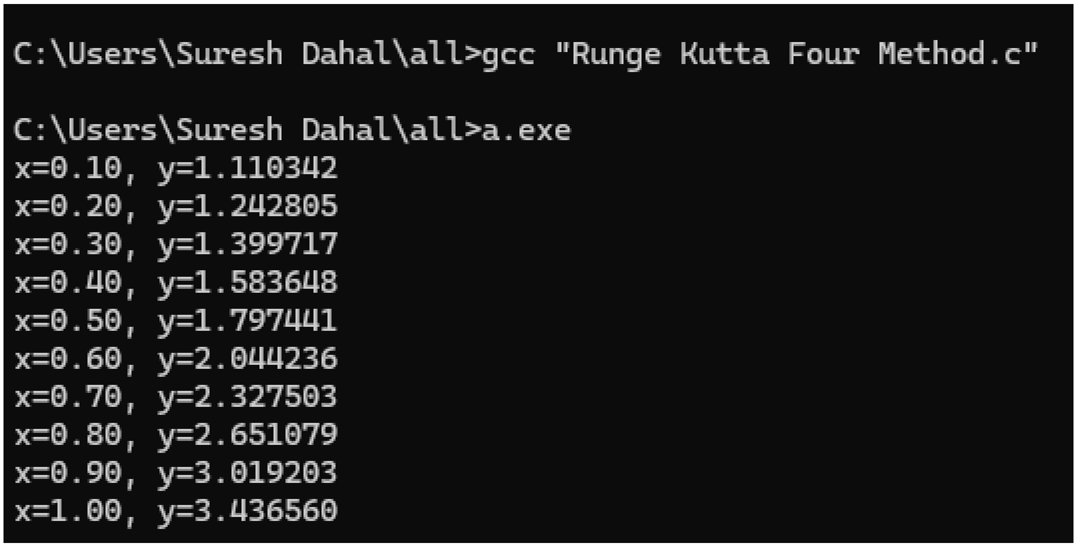
21. Heun’s Method



```
C:\Users\Suresh Dahal\all>gcc "Heuns Method.c"

C:\Users\Suresh Dahal\all>a.exe
x=0.10, y=1.110000
x=0.20, y=1.242050
x=0.30, y=1.398465
x=0.40, y=1.581804
x=0.50, y=1.794894
x=0.60, y=2.040858
x=0.70, y=2.323148
x=0.80, y=2.645578
x=0.90, y=3.012364
x=1.00, y=3.428162
```

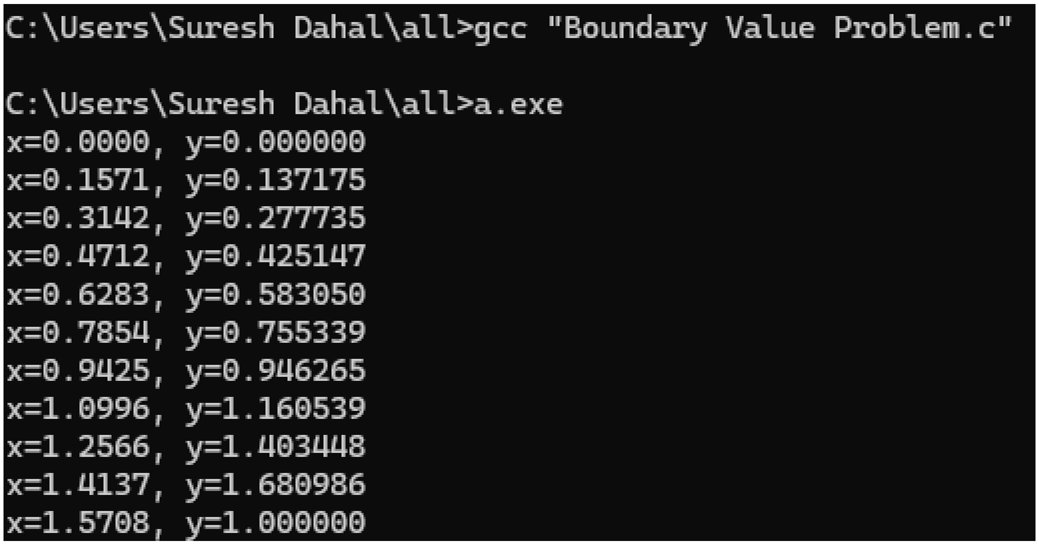
22. **Runge-Kutta Method (4th order)**



```
C:\Users\Suresh Dahal\all>gcc "Runge Kutta Four Method.c"

C:\Users\Suresh Dahal\all>a.exe
x=0.10, y=1.110342
x=0.20, y=1.242805
x=0.30, y=1.399717
x=0.40, y=1.583648
x=0.50, y=1.797441
x=0.60, y=2.044236
x=0.70, y=2.327503
x=0.80, y=2.651079
x=0.90, y=3.019203
x=1.00, y=3.436560
```

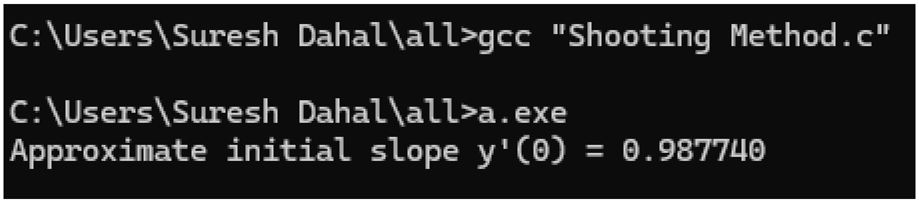
23. **Boundary Value Problem (Using Finite Difference Method)**



```
C:\Users\Suresh Dahal\all>gcc "Boundary Value Problem.c"

C:\Users\Suresh Dahal\all>a.exe
x=0.0000, y=0.000000
x=0.1571, y=0.137175
x=0.3142, y=0.277735
x=0.4712, y=0.425147
x=0.6283, y=0.583050
x=0.7854, y=0.755339
x=0.9425, y=0.946265
x=1.0996, y=1.160539
x=1.2566, y=1.403448
x=1.4137, y=1.680986
x=1.5708, y=1.000000
```

24. **Shooting Method**



```
C:\Users\Suresh Dahal\all>gcc "Shooting Method.c"

C:\Users\Suresh Dahal\all>a.exe
Approximate initial slope y'(0) = 0.987740
```