

Day 1:

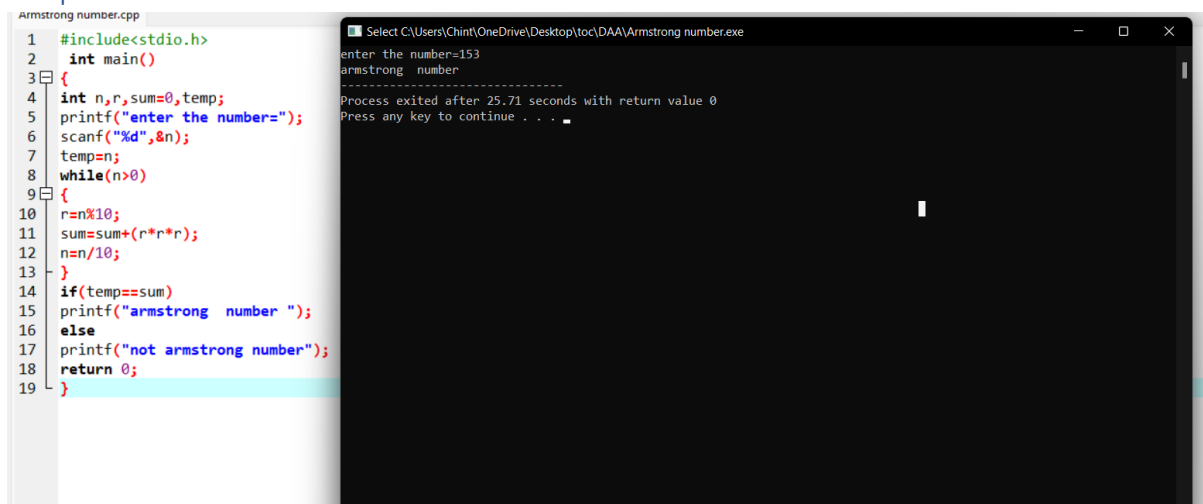
DAA

1. Armstrong number:

Program:

```
1. #include<stdio.h>
2. int main()
3. {
4. int n,r,sum=0,temp;
5. printf("enter the number=");
6. scanf("%d",&n);
7. temp=n;
8. while(n>0)
9. {
10. r=n%10;
11. sum=sum+(r*r*r);
12. n=n/10;
13. }
14. if(temp==sum)
15. printf("armstrong number ");
16. else
17. printf("not armstrong number");
18. return 0;
19. }
```

Output:



The screenshot displays a C++ IDE with two windows. The left window, titled 'Armstrong number.cpp', shows the source code of the program. The right window, titled 'Select C:\Users\Chint\OneDrive\Desktop\tood\DAAD\Armstrong number.exe', shows the program's execution. The output in the right window is as follows:

```
enter the number=153
armstrong number
-----
Process exited after 25.71 seconds with return value 0
Press any key to continue . . .
```

2. Time complexity:

i)

program:

```
#include <stdio.h>
```

```
void function(int min);
```

```
int main()
```

```
{
```

```
int n;
```

```
scanf("%d",&n);
```

```
function(n);
```

```
return 0;
```

```
}
```

```
void function(int n)
```

```
{
```

```
int count=0;
```

```
int i=1,s=1;
```

```
count++;
```

```
count++;
```

```
while(s<=n)
```

```
{
```

```
count++;
```

```
i++;
```

```
count++;
```

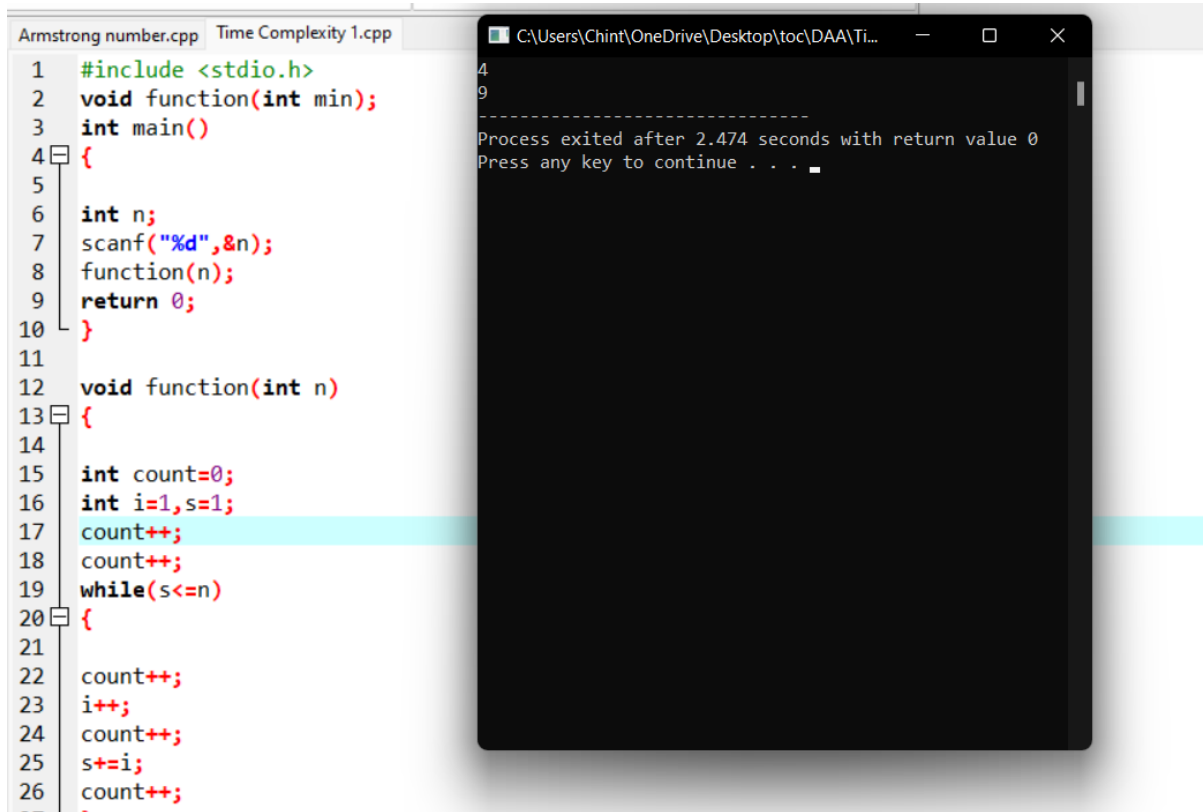
```
s+=i;
```

```
count++;
```

```
}
```

```
count++;  
  
printf("%d",count);  
  
}
```

Output:

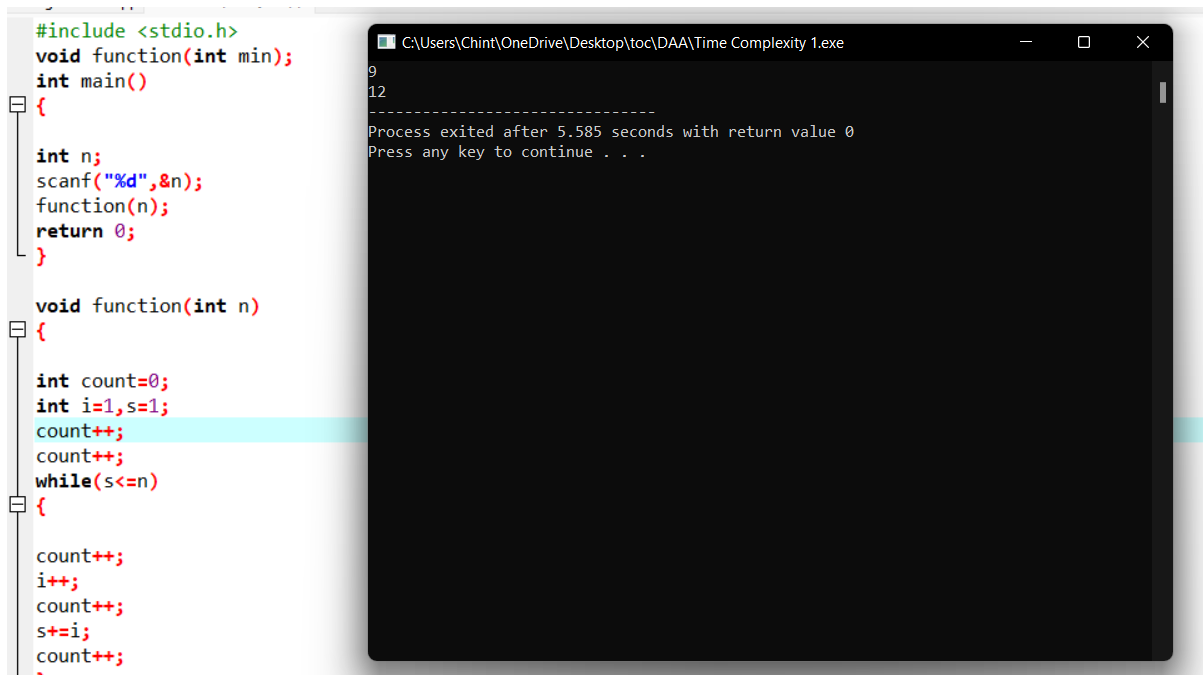


The screenshot shows a C++ IDE with two tabs: 'Armstrong number.cpp' and 'Time Complexity 1.cpp'. The 'Armstrong number.cpp' tab is active, displaying the following code:

```
1 #include <stdio.h>  
2 void function(int min);  
3 int main()  
4 {  
5  
6     int n;  
7     scanf("%d",&n);  
8     function(n);  
9     return 0;  
10 }  
11  
12 void function(int n)  
13 {  
14  
15     int count=0;  
16     int i=1,s=1;  
17     count++;  
18     count++;  
19     while(s<=n)  
20     {  
21  
22         count++;  
23         i++;  
24         count++;  
25         s+=i;  
26         count++;  
27     }
```

The terminal window shows the output of the program:

```
4  
9  
-----  
Process exited after 2.474 seconds with return value 0  
Press any key to continue . . .
```



The screenshot shows a C++ IDE with two tabs: 'Armstrong number.cpp' and 'Time Complexity 1.cpp'. The 'Time Complexity 1.cpp' tab is active, displaying the following code:

```
#include <stdio.h>  
void function(int min);  
int main()  
{  
  
    int n;  
    scanf("%d",&n);  
    function(n);  
    return 0;  
}  
  
void function(int n)  
{  
  
    int count=0;  
    int i=1,s=1;  
    count++;  
    count++;  
    while(s<=n)  
    {  
  
        count++;  
        i++;  
        count++;  
        s+=i;  
        count++;  
    }
```

The terminal window shows the output of the program:

```
9  
12  
-----  
Process exited after 5.585 seconds with return value 0  
Press any key to continue . . .
```

ii)

program:

```
#include <stdio.h>
```

```
void function(int n);
```

```
int main()
```

```
{
```

```
int n; scanf("%d",&n);
```

```
function(n);
```

```
return 0;
```

```
}
```

```
void function(int n)
```

```
{
```

```
int count=0;
```

```
if(n==1)
```

```
{
```

```
count++; count++;
```

```
}
```

```
else
```

```
{
```

```
count++;
```

```
for(int i=1;i<=n;i++)
```

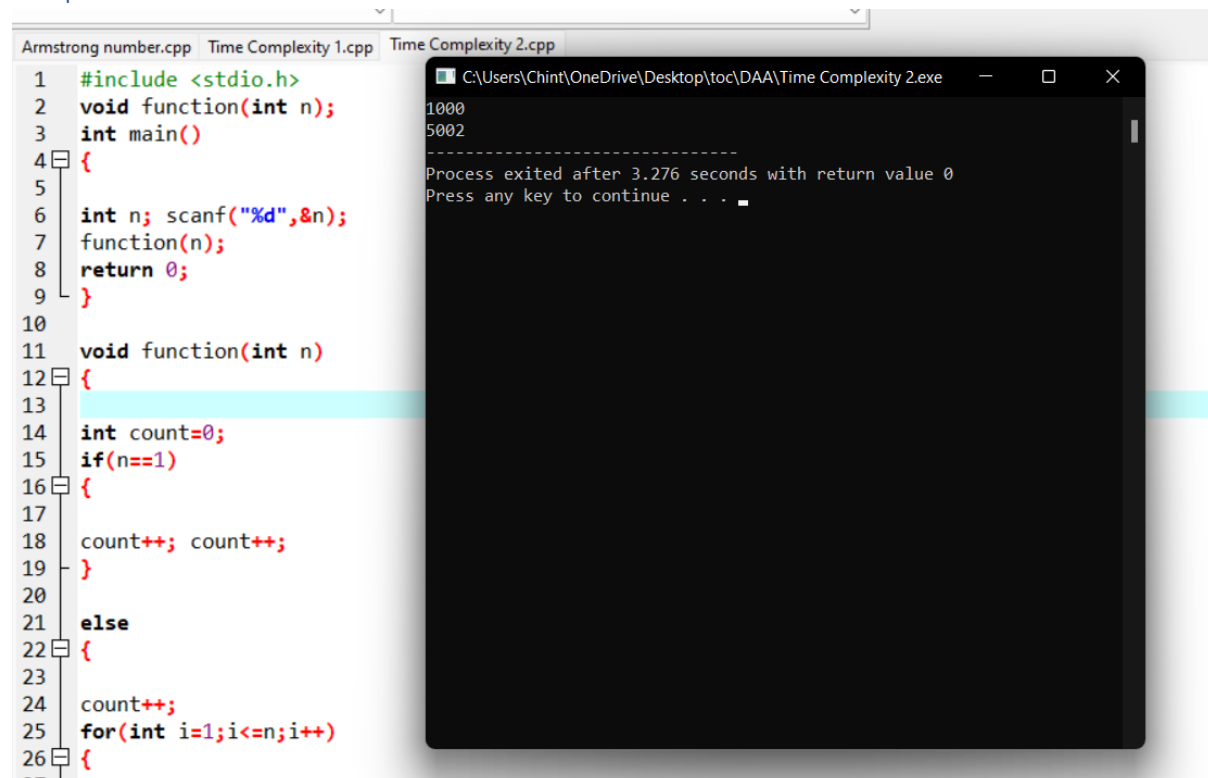
```
{
```

```
count++;
```

```
for(int j=1;j<=n;j++)
```

```
{  
  
count++;  
count++;  
count++;  
count++;  
break;  
}  
  
}  
count++;  
}  
  
printf("%d",count);  
}
```

Output:



The screenshot shows a C++ IDE with two tabs: 'Armstrong number.cpp' and 'Time Complexity 2.cpp'. The 'Armstrong number.cpp' tab is active, displaying the following code:

```
1 #include <stdio.h>
2 void function(int n);
3 int main()
4 {
5
6 int n; scanf("%d",&n);
7 function(n);
8 return 0;
9 }
10
11 void function(int n)
12 {
13
14 int count=0;
15 if(n==1)
16 {
17
18 count++; count++;
19 }
20
21 else
22 {
23
24 count++;
25 for(int i=1;i<=n;i++)
26 {
```

The 'Time Complexity 2.cpp' tab is also visible, showing the following code:

```
1 #include <stdio.h>
2 void function(int n);
3 int main()
4 {
5
6 int n; scanf("%d",&n);
7 function(n);
8 return 0;
9 }
10
11 void function(int n)
12 {
13
14 int count=0;
15 if(n==1)
16 {
17
18 count++; count++;
19 }
20
21 else
22 {
23
24 count++;
25 for(int i=1;i<=n;i++)
26 {
```

The terminal window shows the output of the program:

```
C:\Users\Chint\OneDrive\Desktop\toc\DAA\Time Complexity 2.exe
1000
5002
-----
Process exited after 3.276 seconds with return value 0
Press any key to continue . . .
```

The screenshot shows a C++ IDE with two tabs: 'Armstrong number.cpp' and 'Time Complexity 2.cpp'. The 'Time Complexity 2.cpp' tab is active, displaying the following code:

```
1 #include <stdio.h>
2 void function(int n);
3 int main()
4 {
5
6     int n; scanf("%d",&n);
7     function(n);
8     return 0;
9 }
10
11 void function(int n)
12 {
13
14     int count=0;
15     if(n==1)
16     {
17
18         count++; count++;
19     }
20
21     else
22     {
23
24         count++;
25         for(int i=1;i<=n;i++)
26         {
```

The output window shows the execution results for 'C:\Users\Chint\OneDrive\Desktop\toc\DAA\Time Complexity 2.exe':

```
2
12
-----
Process exited after 18.01 seconds with return value 0
Press any key to continue . . .
```

The screenshot shows the same C++ IDE with the 'Time Complexity 2.cpp' tab active. The code is identical to the previous screenshot. The output window shows the execution results for 'C:\Users\Chint\OneDrive\Desktop\toc\DAA\Time Complexity 2.exe':

```
143
717
-----
Process exited after 3.398 seconds with return value 0
Press any key to continue . . .
```

iii)

program:

```
#include <stdio.h>
```

```
int factor(int n);
```

```
int count=0;
```

```
int main()
```

```
{
```

```
int n;
```

```
scanf("%d",&n);
```

```
factor(n);
```

```
printf("%d",count);
```

```
return 0;
```

```
}
```

```
int factor(int n)
```

```
{
```

```
int i; count++;
```

```
for(i=1;i<=n;++i)
```

```
{
```

```
count++;
```

```
if(n%i==0)
```

```
{
```

```
//print
```

```
}count++;
```

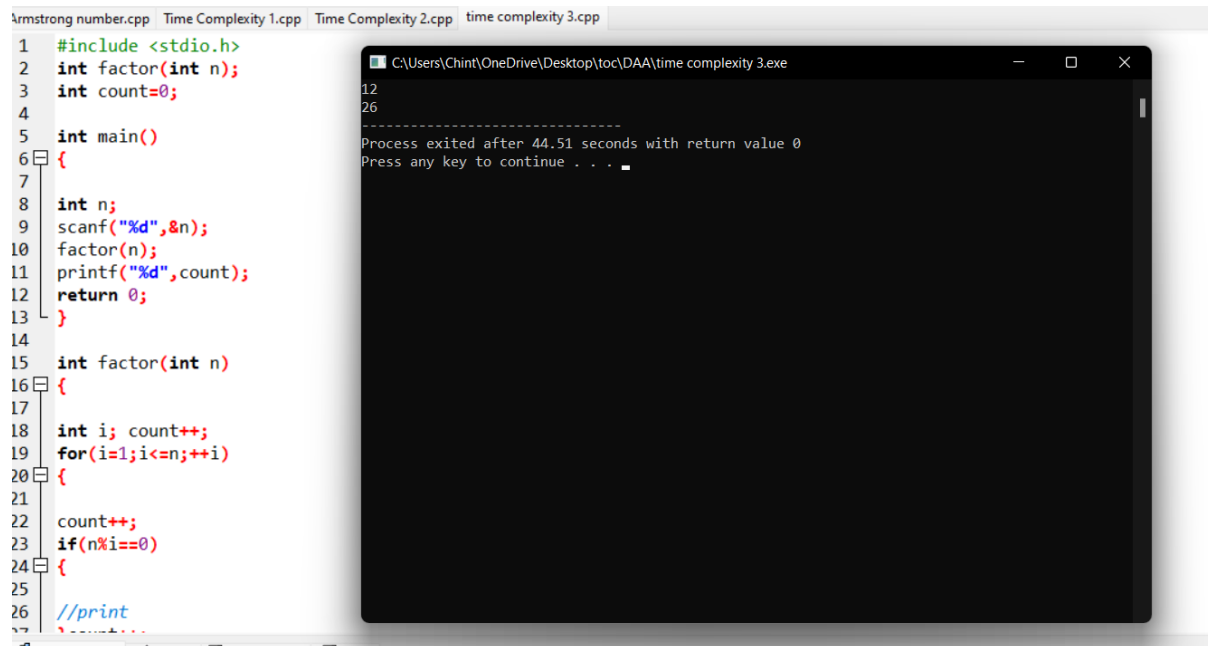
```
}
```

```
count++;
```

```
return 0;
```

```
}
```

Output:



The screenshot shows a C++ IDE with two windows. The left window, titled 'time complexity 3.cpp', contains the following code:

```
1 #include <stdio.h>
2 int factor(int n);
3 int count=0;
4
5 int main()
6 {
7
8     int n;
9     scanf("%d",&n);
10    factor(n);
11    printf("%d",count);
12    return 0;
13 }
14
15 int factor(int n)
16 {
17
18     int i; count++;
19     for(i=1;i<=n;++i)
20     {
21         count++;
22         if(n%i==0)
23         {
24             //print
25         }
26     }
27 }
```

The right window, titled 'C:\Users\Chint\OneDrive\Desktop\toc\DAAtime complexity 3.exe', shows the program's execution output:

```
12
26
-----
Process exited after 44.51 seconds with return value 0
Press any key to continue . . .
```

iv)

program:

```
#include <stdio.h>
```

```
void function(int n);
```

```
int main()
```

```
{
```

```
int n;
```

```
scanf("%d",&n);
```

```
function(n);
```

```
return 0;
```

```
}
```

```
void function(int n)
```

```
{
```

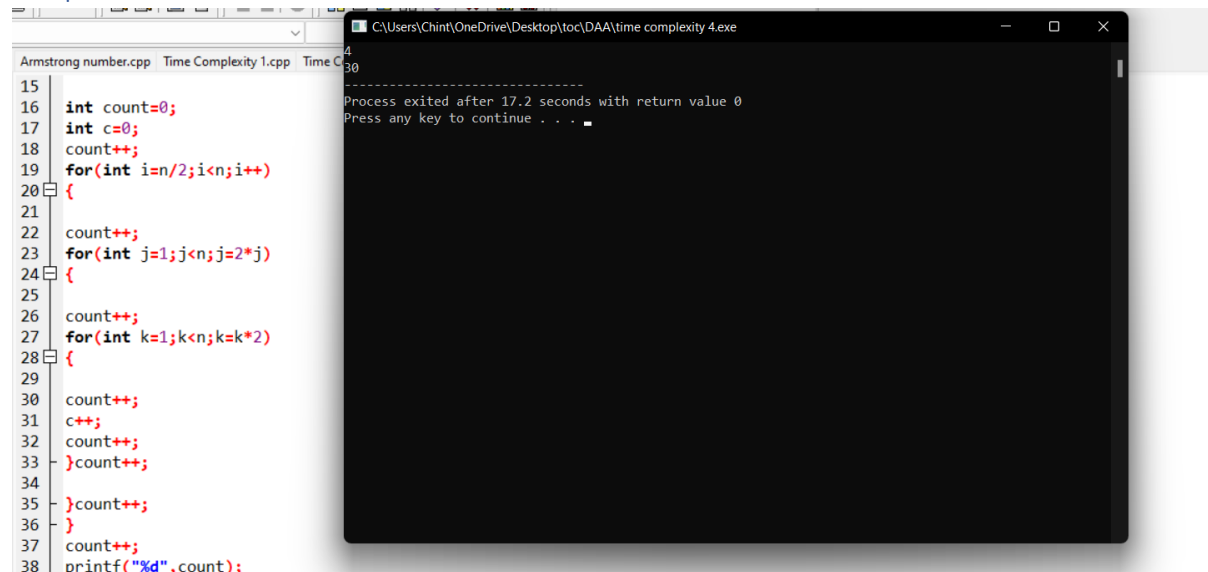
```
int count=0;
```

```
int c=0;
```



```
count++;  
for(int i=n/2;i<n;i++)  
{  
  
count++;  
for(int j=1;j<n;j=2*j)  
{  
  
count++;  
for(int k=1;k<n;k=k*2)  
{  
  
count++;  
c++;  
count++;  
}count++;  
  
}count++;  
}  
count++;  
printf("%d",count);  
}
```

Output:



The screenshot shows a C++ IDE with two files: 'Armstrong number.cpp' and 'Time Complexity 1.cpp'. The 'Armstrong number.cpp' file contains the following code:

```
15 int count=0;
16 int c=0;
17 count++;
18 for(int i=n/2;i<n;i++)
19 {
20     count++;
21     for(int j=1;j<n;j=2*j)
22     {
23         count++;
24         for(int k=1;k<n;k=k*2)
25         {
26             count++;
27             c++;
28             count++;
29         }count++;
30     }count++;
31 }count++;
32 printf("%d",count);
```

The terminal window shows the output of the program:

```
Process exited after 17.2 seconds with return value 0
Press any key to continue . . .
```

v)

program:

```
#include <stdio.h>
```

```
void reverse(int n);
```

```
int main()
```

```
{
```

```
int n;
```

```
scanf("%d",&n);
```

```
reverse(n);
```

```
return 0;
```

```
}
```

```
void reverse(int n)
```

```
{
```

```
int count=0;
```

```
int rev=0,
```

```
remainder;
```

```
count++;
```

```
while(n!=0)

{

count++;

remainder=n%10;

count++;

rev=rev*10+remainder;

count++;

n=n/10;

count++;

}

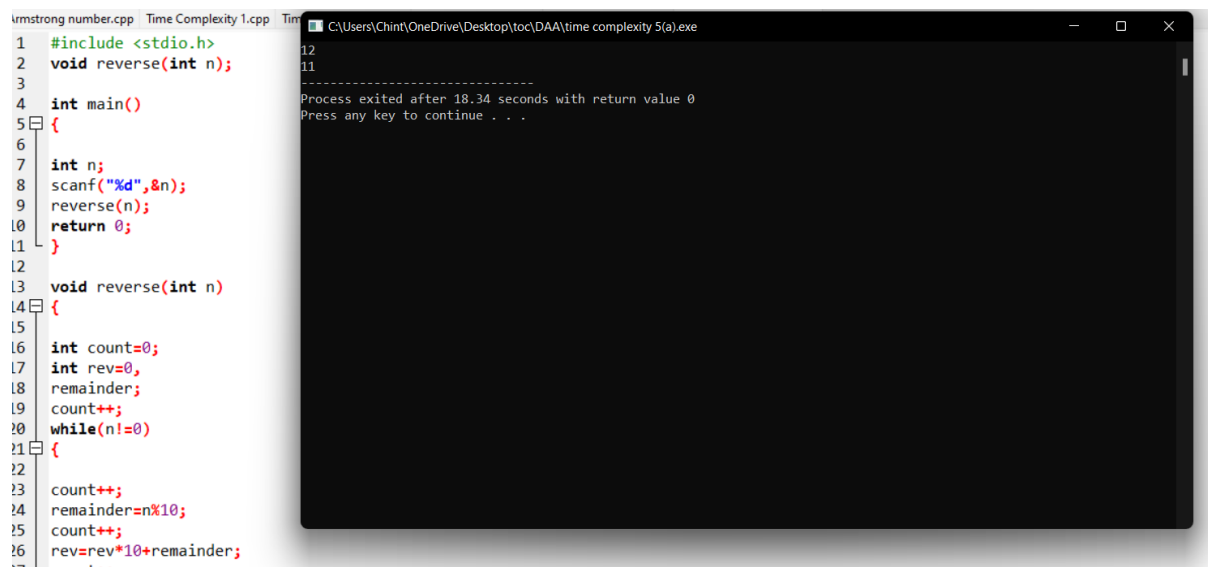
count++;

count++;

printf("%d",count);

}
```

Output:



```
1 #include <stdio.h>
2 void reverse(int n);
3
4 int main()
5 {
6
7     int n;
8     scanf("%d",&n);
9     reverse(n);
10    return 0;
11 }
12
13 void reverse(int n)
14 {
15
16     int count=0;
17     int rev=0;
18     remainder;
19     count++;
20     while(n!=0)
21     {
22
23         count++;
24         remainder=n%10;
25         count++;
26         rev=rev*10+remainder;
27     }
```

3.Binary search:

Program:

```
#include<stdio.h>
```

```
int main()
```

```
{
    int c=0;

    int n,k,i,low,high,mid,a[50],temp;

    printf("Enter number of elements:");

    scanf("%d",&n);

    printf("Enter elements:\n");

    for(i=0;i<n;i++)
    {
        c++;

        scanf("%d",&a[i]);
    }

    c++;

    printf("Enter Element to search:");

    scanf("%d",&k);

    low=0; c++;

    high=n-1; c++;

    mid=low+high/2; c++;

    c++;

    while(low<=high)
    {
        c++;

        c++;

        if(a[mid]<k)
        {
            low=mid+1; c++;
        }

        else if(a[mid]==k)
        {
            printf("\nElement is found at index %d\n",mid);

            break;
        }
    }
```

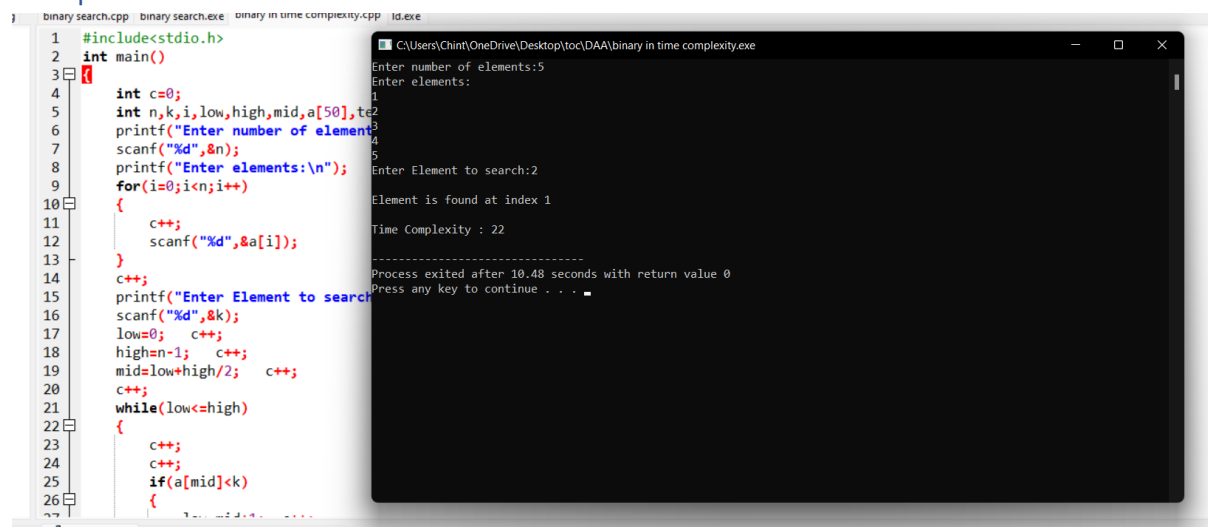
```
else
{
    high=mid-1; c++;
}

mid=(low+high)/2; c++;
}

c++;
c++;
if(low>high)
{
    printf("Element is not found\n");
}

printf("\nTime Complexity : %d\n",c);
}
```

Output:



```
1 #include<stdio.h>
2 int main()
3 {
4     int c=0;
5     int n,k,i,low,high,mid,a[50],t;
6     printf("Enter number of elements:");
7     scanf("%d",&n);
8     printf("Enter elements:\n");
9     for(i=0;i<n;i++)
10    {
11        c++;
12        scanf("%d",&a[i]);
13    }
14    c++;
15    printf("Enter Element to search:");
16    scanf("%d",&k);
17    low=0; c++;
18    high=n-1; c++;
19    mid=low+high/2; c++;
20    c++;
21    while(low<=high)
22    {
23        c++;
24        c++;
25        if(a[mid]<k)
26        {
27            low=mid+1; c++;
28        }
29        else
30        {
31            high=mid-1; c++;
32        }
33        mid=(low+high)/2; c++;
34    }
35    if(low>high)
36    {
37        printf("Element is not found\n");
38    }
39    printf("\nTime Complexity : %d\n",c);
40 }
```

```
C:\Users\Chint\OneDrive\Desktop\toc\DAA\binary in time complexity.exe
Enter number of elements:5
Enter elements:
1
2
3
4
5
Enter Element to search:2
Element is found at index 1
Time Complexity : 22
-----
Process exited after 10.48 seconds with return value 0
Press any key to continue . . .
```

4.Linear search:

Program:

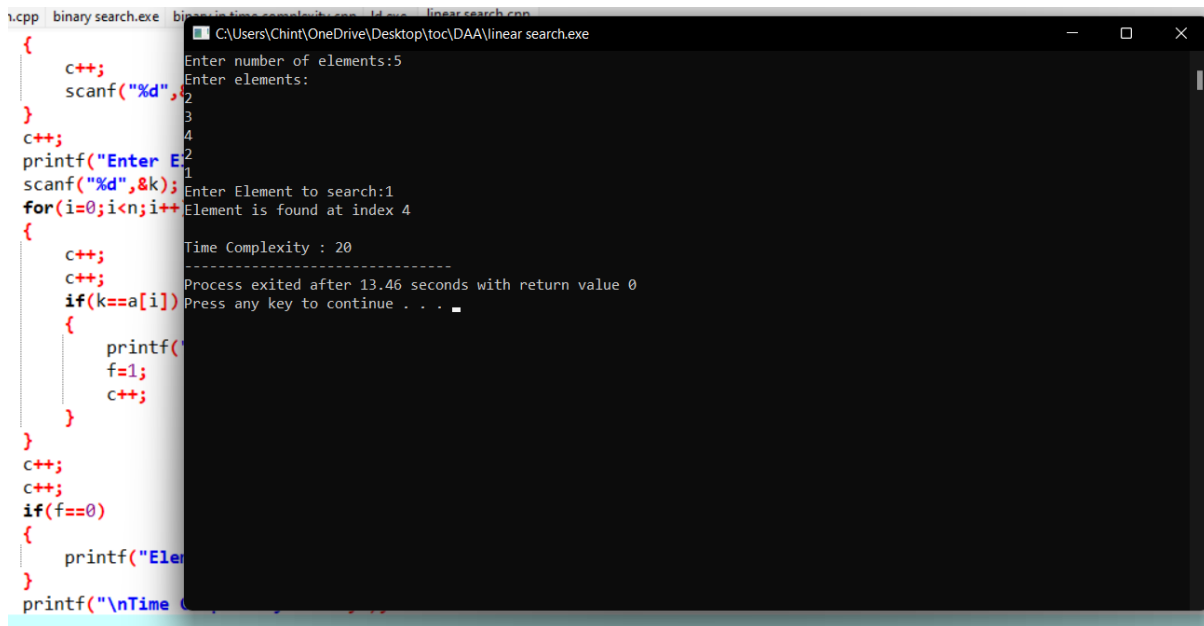
```
#include<stdio.h>

int main()
{
    int c=0;

    int n,k,i,j,f=0,a[50];
```

```
c++;  
  
printf("Enter number of elements:");  
  
scanf("%d",&n);  
  
printf("Enter elements:\n");  
  
for(i=0;i<n;i++)  
{  
  
    c++;  
  
    scanf("%d",&a[i]);  
  
}  
  
c++;  
  
printf("Enter Element to search:");  
  
scanf("%d",&k);  
  
for(i=0;i<n;i++)  
{  
  
    c++;  
  
    c++;  
  
    if(k==a[i])  
    {  
  
        printf("Element is found at index %d\n",i);  
  
        f=1;  
  
        c++;  
  
    }  
  
}  
  
c++;  
  
c++;  
  
if(f==0)  
{  
  
    printf("Element is not found");  
  
}  
  
printf("\nTime Complexity : %d",c);  
  
}
```

Output:



```
1.cpp binary search.exe C:\Users\Chint\OneDrive\Desktop\toc\DAA\linear search.exe
{
    c++;
    scanf("%d", &n);
}
c++;
printf("Enter Elements:");
scanf("%d", &k);
for(i=0; i<n; i++)
{
    c++;
    if(k==a[i])
    {
        printf("Element is found at index %d", i);
        f=1;
        c++;
    }
}
c++;
c++;
if(f==0)
{
    printf("Element not found");
}
printf("\nTime Complexity : %d", c);
```

5.Reverse a number:

Program:

```
#include <stdio.h>
```

```
int main() {
```

```
    int n, reverse = 0, remainder;
```

```
    printf("Enter an integer: ");
```

```
    scanf("%d", &n);
```

```
    while (n != 0) {
```

```
        remainder = n % 10;
```

```
        reverse = reverse * 10 + remainder;
```

```
        n /= 10;
```

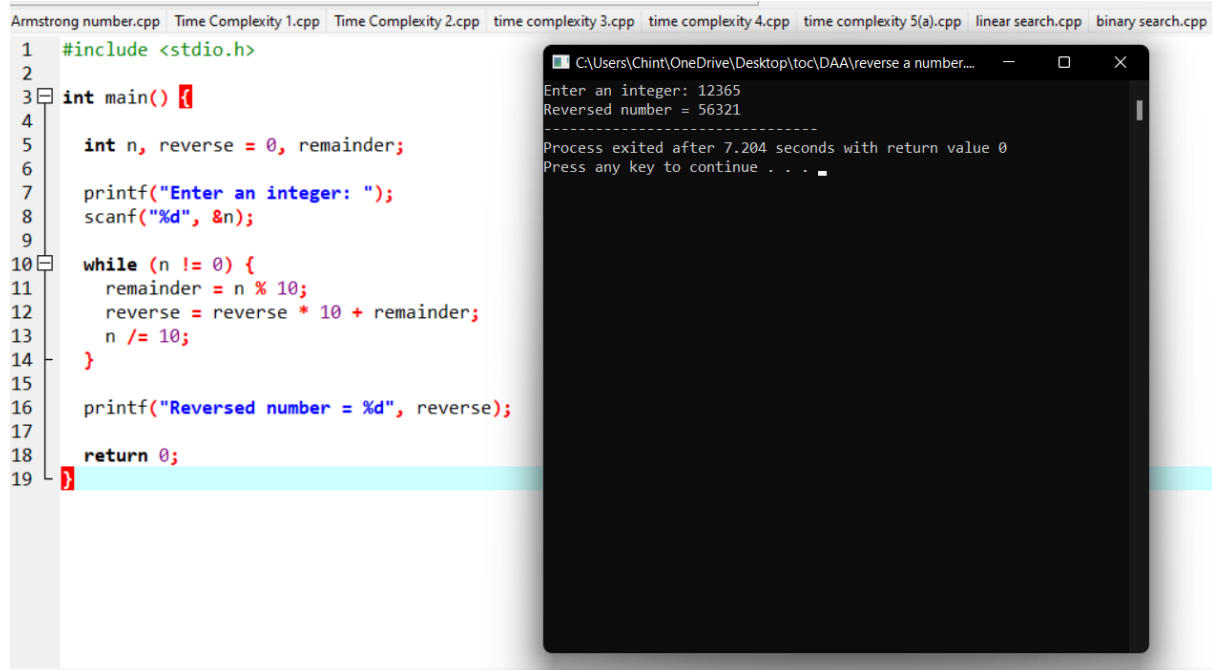
```
    }
```

```
printf("Reversed number = %d", reverse);
```

```
return 0;
```

```
}
```

Output:



The screenshot shows a C++ IDE with a file explorer at the top listing several files: Armstrong number.cpp, Time Complexity 1.cpp, Time Complexity 2.cpp, time complexity 3.cpp, time complexity 4.cpp, time complexity 5(a).cpp, linear search.cpp, and binary search.cpp. The main editor displays the following code:

```
1 #include <stdio.h>
2
3 int main() {
4     int n, reverse = 0, remainder;
5
6     printf("Enter an integer: ");
7     scanf("%d", &n);
8
9     while (n != 0) {
10         remainder = n % 10;
11         reverse = reverse * 10 + remainder;
12         n /= 10;
13     }
14
15     printf("Reversed number = %d", reverse);
16
17     return 0;
18 }
```

Overlaid on the right is a terminal window titled "C:\Users\Chint\OneDrive\Desktop\toc\DAA\reverse a number...". It shows the execution output:

```
Enter an integer: 12365
Reversed number = 56321
-----
Process exited after 7.204 seconds with return value 0
Press any key to continue . . .
```

6.Matrix multiplication:

Program:

```
#include<stdio.h>
```

```
int main(){
```

```
int a[2][2], b[2][2], c[2][2], i, j, count=0;
```

```
int m1, m2, m3, m4 , m5, m6, m7;
```

```
printf("Enter the 4 elements of first matrix:");
```

```
count++;
```

```
for(i = 0; i < 2; i++)
```

```
{
```

```
count++;
```

```
for(j = 0; j < 2; j++)
```

```
{
```



```
count++;

scanf("%d", &a[i][j]);

}

}

count++;

count++;


printf("Enter the 4 elements of second matrix: ");

for(i = 0; i < 2; i++)

{

count++;

for(j = 0; j < 2; j++)

{

count++;

scanf("%d", &b[i][j]);

}

}

count++;

count++;


printf("\nThe first matrix is\n");

for(i = 0; i < 2; i++){

count++;

printf("\n");

for(j = 0; j < 2; j++){

count++;

printf("%d\t", a[i][j]);

}

}

count++;

count++;
```

```
printf("\nThe second matrix is\n");

    for(i = 0; i < 2; i++){

        count++;

        printf("\n");

        for(j = 0; j < 2; j++){

            count++;

            printf("%d\t", b[i][j]);

        }

    }

count++;

    count++;

m1= (a[0][0] + a[1][1]) * (b[0][0] + b[1][1]);

count++;

m2= (a[1][0] + a[1][1]) * b[0][0];

count++;

m3= a[0][0] * (b[0][1] - b[1][1]);

count++;

m4= a[1][1] * (b[1][0] - b[0][0]);

count++;

m5= (a[0][0] + a[0][1]) * b[1][1];

count++;

m6= (a[1][0] - a[0][0]) * (b[0][0]+b[0][1]);

count++;

    m7= (a[0][1] - a[1][1]) * (b[1][0]+b[1][1]);

    count++;


c[0][0] = m1 + m4- m5 + m7;

count++;

    c[0][1] = m3 + m5;

    count++;
```

```
c[1][0] = m2 + m4;

count++;

c[1][1] = m1 - m2 + m3 + m6;

count++;


printf("\nAfter multiplication using Strassen's algorithm \n");

for(i = 0; i < 2 ; i++){

    count++;

    printf("\n");

    for(j = 0;j < 2; j++){

        count++;

        printf("%d\t", c[i][j]);

    }

}

count++;

count++;


printf(" time complexity is %d",count);


return 0;

}

}
```

Output;

```
61 m4= a[1][1] *
62 count++;
63 m5= (a[0][0]
64 count++;
65 m6= (a[1][0]
66 count++;
67 m7= (a[0]
68 count++;
69
70 c[0][0] = m1
71 count++;
72 c[0][1] =
73 count++;
74 c[1][0] =
75 count++;
76 c[1][1] =
77 count++;
78
79 printf("
80 for(i = 0
81 count++;
82 printf(
83 for(j = 0
84 count++;
85
86 }
```

Enter the 4 elements of first matrix: 5
1
2
3
4
Enter the 4 elements of second matrix: 3
4
5
6
7
The first matrix is
5 1
1 2
The second matrix is
3 4
5 6
After multiplication using Strassen's algorithm
20 26
13 16 time complexity is 52
Process exited after 15.24 seconds with return value 0
Press any key to continue . . .

Compilation results...
- Errors: 0
- Warnings: 0
- Output Filename: C:\Users\Chint\OneDrive\Desktop\too\DAA\strassen's multiplication.exe
- Output Size: 129.97265625 KiB
- Compilation Time: 0.28s

7.Prime

Program:

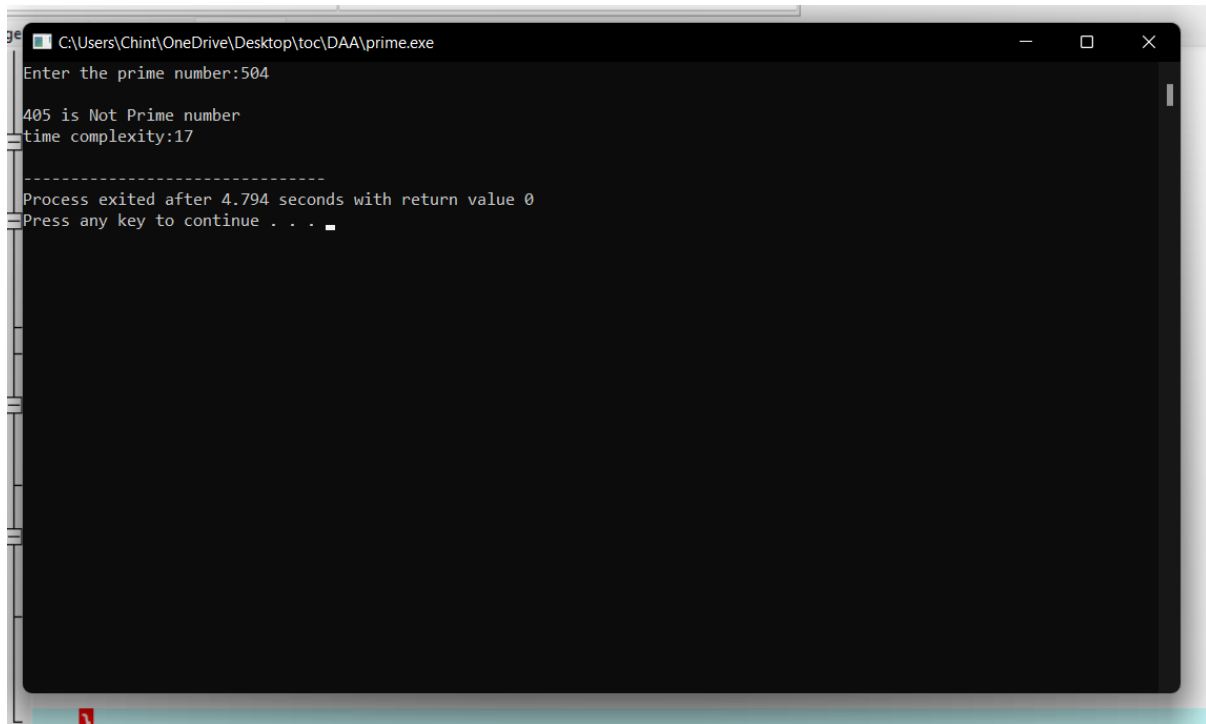
```
#include <stdio.h>
```

```
int main()
{
    int count=0;
    int n, reverse,sum=0 , flag;
    printf("Enter the prime number:");
    scanf("%d",&n);
    while(n!=0)
    {
        count++;
        reverse = n%10;
        count++;
        sum = sum*10 + reverse;
        count++;
        n= n/10;
        count++;
    }
```

```
}

    printf("\n");
flag = 0;
for (int j = 2; j <= sum / 2; j++)
{
    count++;
    if ((sum % j) == 0)
    {
        count++;
        flag = 1;
        break;
    }
}
if (flag == 0)
{
    count++;
    printf("%d is also prime number",sum);
}
else
{
    count++;
    printf("%d is Not Prime number\n",sum);
}
count++;
printf("time complexity:%d\n",count);
}
```

Output:



```
C:\Users\Chint\OneDrive\Desktop\loc\DAA\prime.exe
Enter the prime number:504
405 is Not Prime number
time complexity:17
-----
Process exited after 4.794 seconds with return value 0
Press any key to continue . . .
```

8.GCD:

Program:

```
#include <stdio.h>

int main()
{
    int n1, n2, i, GCD_Num;

    int count=0;

    printf ( " Enter any two numbers: \n ");
    scanf ( "%d %d", &n1, &n2);

    for( i = 1; i <= n1 && i <= n2; ++i)
    {
        count++;

        if (n1 % i ==0 && n2 % i == 0)
            GCD_Num = i;

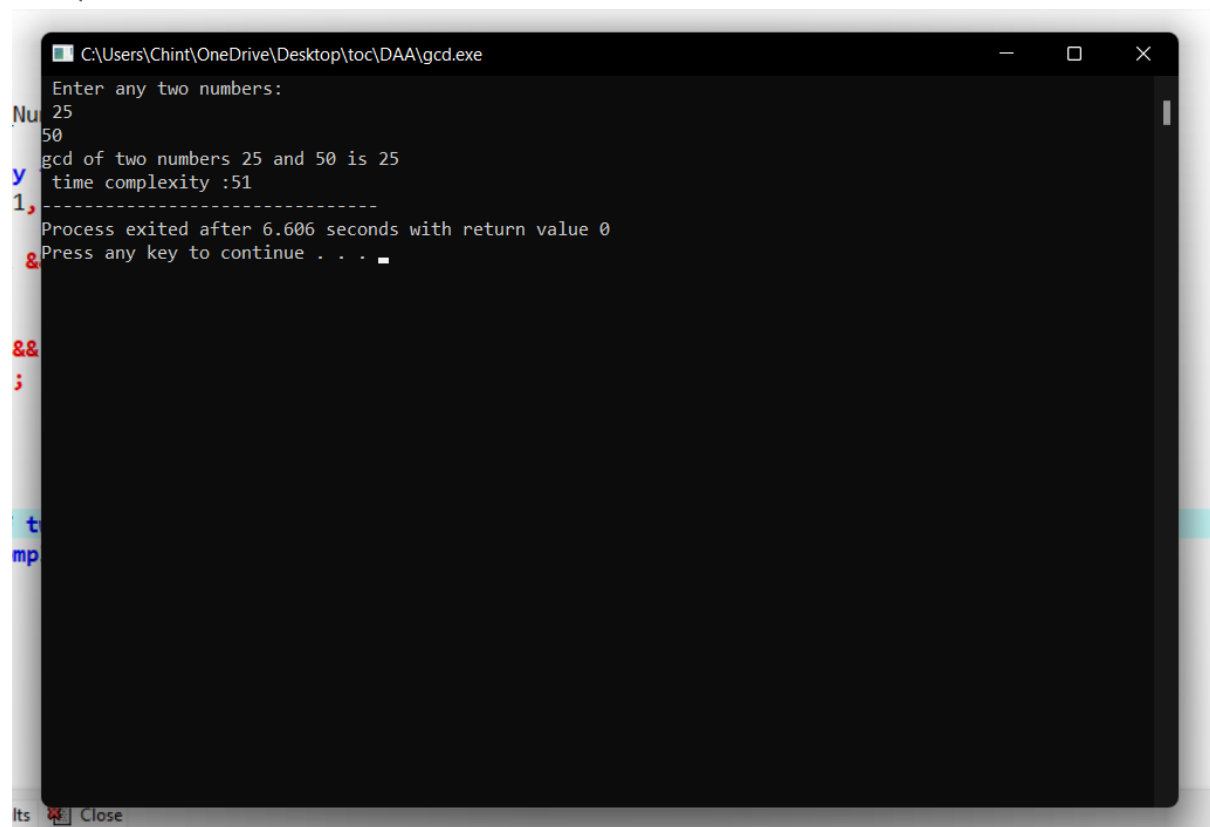
        count++;
    }
}
```

```
        count++;

    printf("gcd of two numbers %d and %d is %d \n ", n1, n2, GCD_Num);
    printf("time complexity :%d ",count);

    return 0;
}
```

Output:



```
C:\Users\Chint\OneDrive\Desktop\toc\DAA\gcd.exe
Enter any two numbers:
25
50
gcd of two numbers 25 and 50 is 25
time complexity :51
-----
Process exited after 6.606 seconds with return value 0
Press any key to continue . . .
1
```

9.Pascal triangle:

Program:

```
#include<stdio.h>

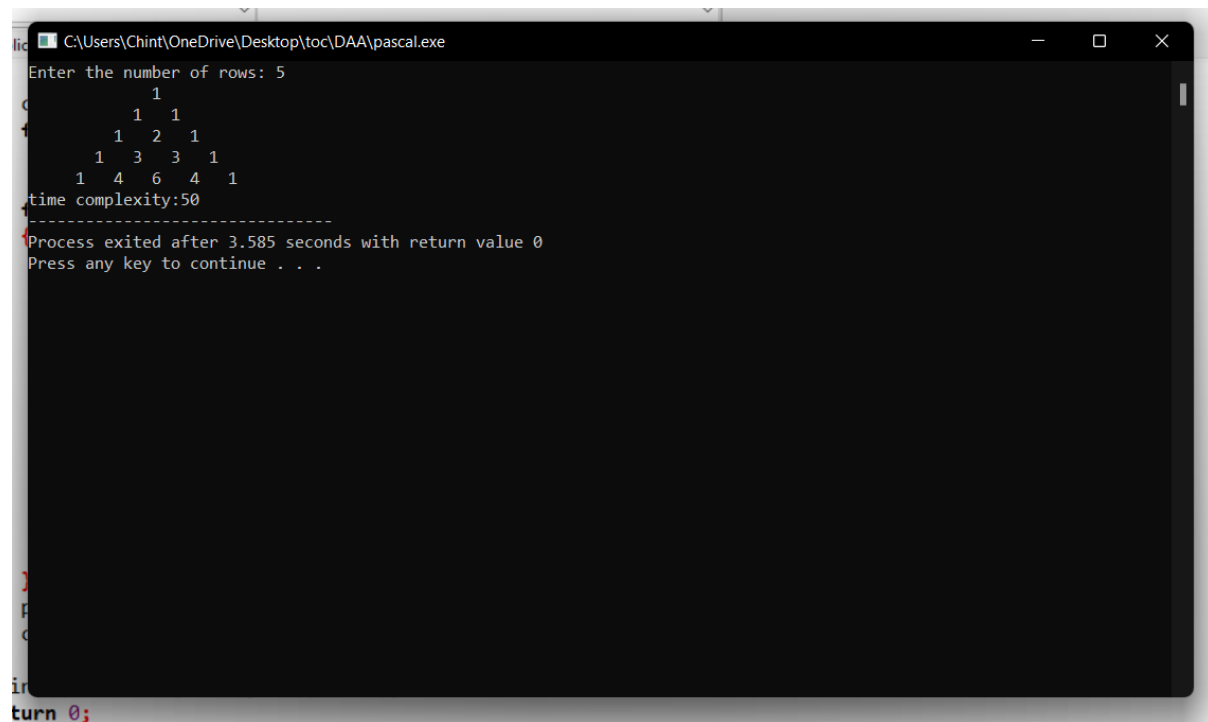
int main()
{
    int rows, coef = 1, space, i, j;

    int count=0;

    printf("Enter the number of rows: ");
```

```
scanf("%d", &rows);  
for (i = 0; i < rows; i++)  
{  
    count++;  
    for (space = 1; space <= rows - i; space++)  
        printf(" ");  
    count++;  
    for (j = 0; j <= i; j++)  
    {  
        count++;  
        if(j == 0 || i == 0){  
            coef = 1;  
            count++;  
        }  
  
        else  
        {  
            coef = coef * (i - j + 1) / j;  
        }  
        count++;  
        printf("%4d", coef);  
    }  
    printf("\n");  
    count++;  
}  
printf("time complexity:%d",count);  
return 0;  
}
```


Output:



```
C:\Users\Chint\OneDrive\Desktop\toe\DAA\pascal.exe
Enter the number of rows: 5
      1
     1 1
    1 2 1
   1 3 3 1
  1 4 6 4 1
time complexity:50
-----
Process exited after 3.585 seconds with return value 0
Press any key to continue . . .
```

10.Largest number:

Program:

```
#include <stdio.h>

int main() {

    int n;

    int count=0;

    double arr[100];

    printf("Enter the number of elements (1 to 100): ");

    scanf("%d", &n);

    count++;

    for (int i = 0; i < n; ++i) {

        count++;

        printf("Enter number%d: ", i + 1);

        scanf("%lf", &arr[i]);

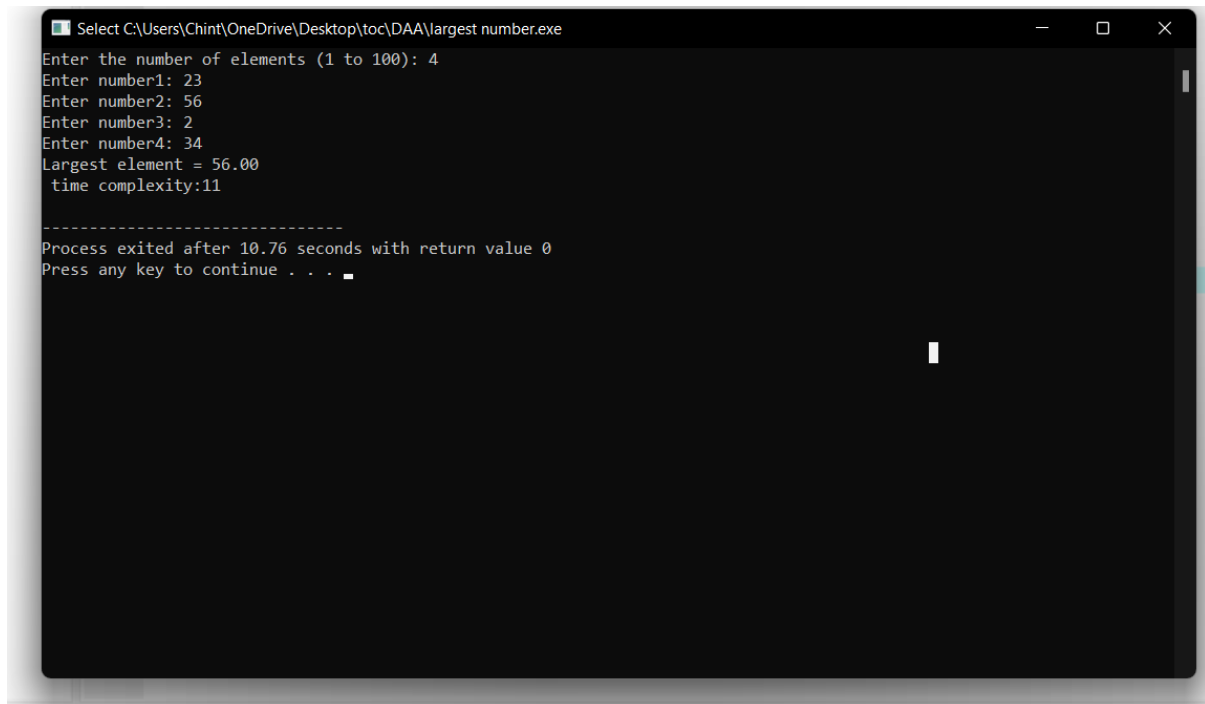
    }

    for (int i = 1; i < n; ++i) {

        count++;
```

```
if (arr[0] < arr[i]) {  
    arr[0] = arr[i];  
}  
count++;  
}  
  
printf("Largest element = %.2lf \n ", arr[0]);  
printf("time complexity:%d\n",count);  
return 0;  
}
```

Output:



The screenshot shows a Windows command prompt window titled "Select C:\Users\Chint\OneDrive\Desktop\toc\DAA\largest number.exe". The user enters the number of elements (1 to 100) as 4. Then, they enter four numbers: 23, 56, 2, and 34. The program outputs "Largest element = 56.00" and "time complexity:11". Below this, it shows "Process exited after 10.76 seconds with return value 0" and "Press any key to continue . . .".

```
Select C:\Users\Chint\OneDrive\Desktop\toc\DAA\largest number.exe  
Enter the number of elements (1 to 100): 4  
Enter number1: 23  
Enter number2: 56  
Enter number3: 2  
Enter number4: 34  
Largest element = 56.00  
time complexity:11  
  
-----  
Process exited after 10.76 seconds with return value 0  
Press any key to continue . . .
```

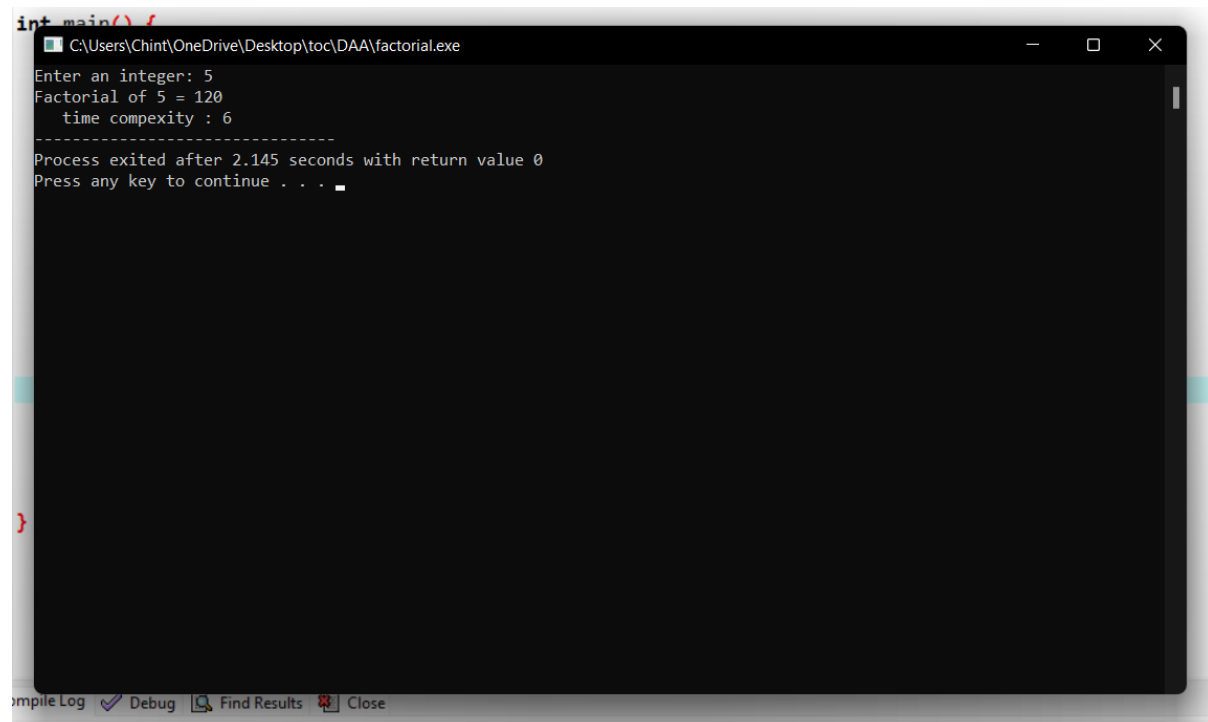
11.Factorial:

Program:

```
#include <stdio.h>  
  
int main() {  
    int n, i;  
    int count=0;  
    unsigned long long fact = 1;
```

```
printf("Enter an integer: ");  
scanf("%d", &n);  
count++;  
if (n < 0)  
    printf("Error! Factorial of a negative number doesn't exist.");  
else {  
    for (i = 1; i <= n; ++i) {  
        fact *= i;  
        count++;  
    }  
    printf("Factorial of %d = %llu \n ", n, fact);  
    printf(" time compexity : %d ",count);  
}  
  
return 0;  
}
```

Output:



```
int main() {  
C:\Users\Chint\OneDrive\Desktop\toc\DAA\factorial.exe  
Enter an integer: 5  
Factorial of 5 = 120  
    time compexity : 6  
-----  
Process exited after 2.145 seconds with return value 0  
Press any key to continue . . .  
}
```

12.Perfect numbers:

Program:

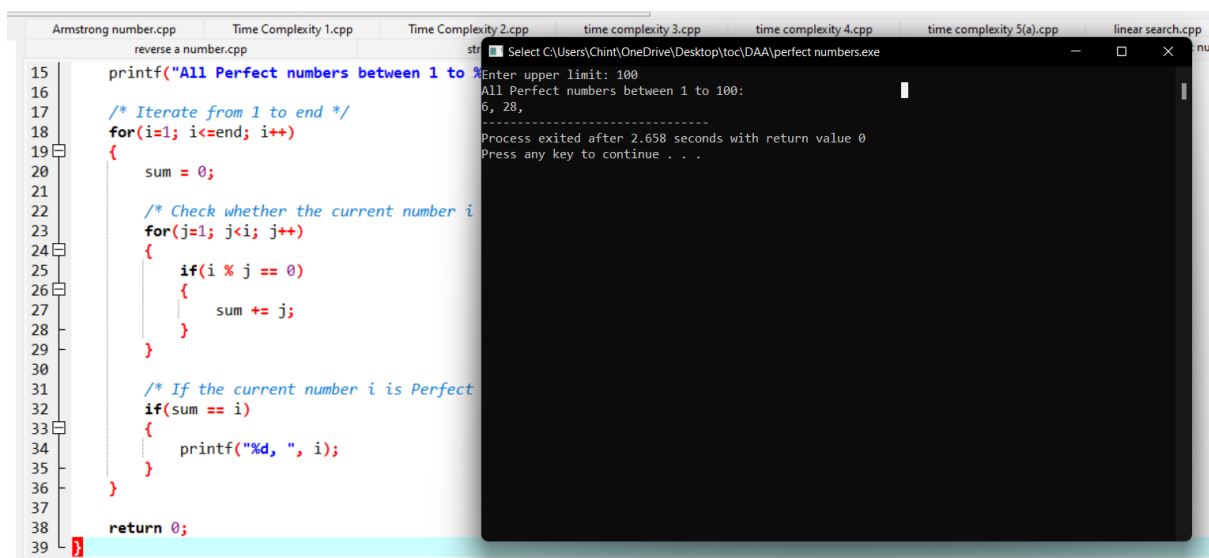
```
/**  
 * C program to print all Perfect numbers between 1 to n  
 */  
  
#include <stdio.h>  
  
int main()  
{  
    int i, j, end, sum;  
  
    /* Input upper limit to print perfect number */  
    printf("Enter upper limit: ");  
    scanf("%d", &end);  
  
    printf("All Perfect numbers between 1 to %d:\n", end);  
  
    /* Iterate from 1 to end */  
    for(i=1; i<=end; i++)  
    {  
        sum = 0;  
  
        /* Check whether the current number i is Perfect number or not */  
        for(j=1; j<i; j++)  
        {  
            if(i % j == 0)  
            {  
                sum += j;  
            }  
        }  
    }  
}
```

```
    }

    /* If the current number i is Perfect number */
    if(sum == i)
    {
        printf("%d, ", i);
    }
}

return 0;
}
```

Output:



The screenshot shows a C++ IDE with multiple tabs. The active tab is 'reverse a number.cpp', which contains the code for finding perfect numbers. The code iterates from 1 to 100, checks if each number is perfect by summing its divisors, and prints the perfect numbers found. The output window shows the execution results: 'Enter upper limit: 100', 'All Perfect numbers between 1 to 100: 6, 28,', and 'Process exited after 2.658 seconds with return value 0'. The output window also prompts the user to 'Press any key to continue ...'.

```
15 printf("All Perfect numbers between 1 to 100: ");
16
17 /* Iterate from 1 to end */
18 for(i=1; i<=end; i++)
19 {
20     sum = 0;
21
22     /* Check whether the current number i is Perfect number */
23     for(j=1; j<i; j++)
24     {
25         if(i % j == 0)
26         {
27             sum += j;
28         }
29     }
30
31     /* If the current number i is Perfect number */
32     if(sum == i)
33     {
34         printf("%d, ", i);
35     }
36 }
37
38 return 0;
39
```

str Select C:\Users\Chint\OneDrive\Desktop\toC\DA\perfect numbers.exe

Enter upper limit: 100

All Perfect numbers between 1 to 100: 6, 28,

Process exited after 2.658 seconds with return value 0

Press any key to continue . . .

13. Palindrome:

Program:

```
#include<stdio.h>
```

```
int main()
```

```
{
```

```
    int i,n,r,s=0;
```

```
    printf("\n Enter Integer Number:");
```

```
scanf("%d",&n);
```

```
//LOOP TO FIND REVERSE OF A NUMBER
```

```
for(i=n;i>0; )
```

```
{
```

```
    r=i%10;
```

```
    s=s*10+r;
```

```
    i=i/10;
```

```
}
```

```
/* CHECKING IF THE NUMBER ENTERED AND THE REVERSE NUMBER IS EQUAL OR NOT */
```

```
if(s==n)
```

```
{
```

```
    printf("\n %d is a Palindrome Number",n);
```

```
}
```

```
else
```

```
{
```

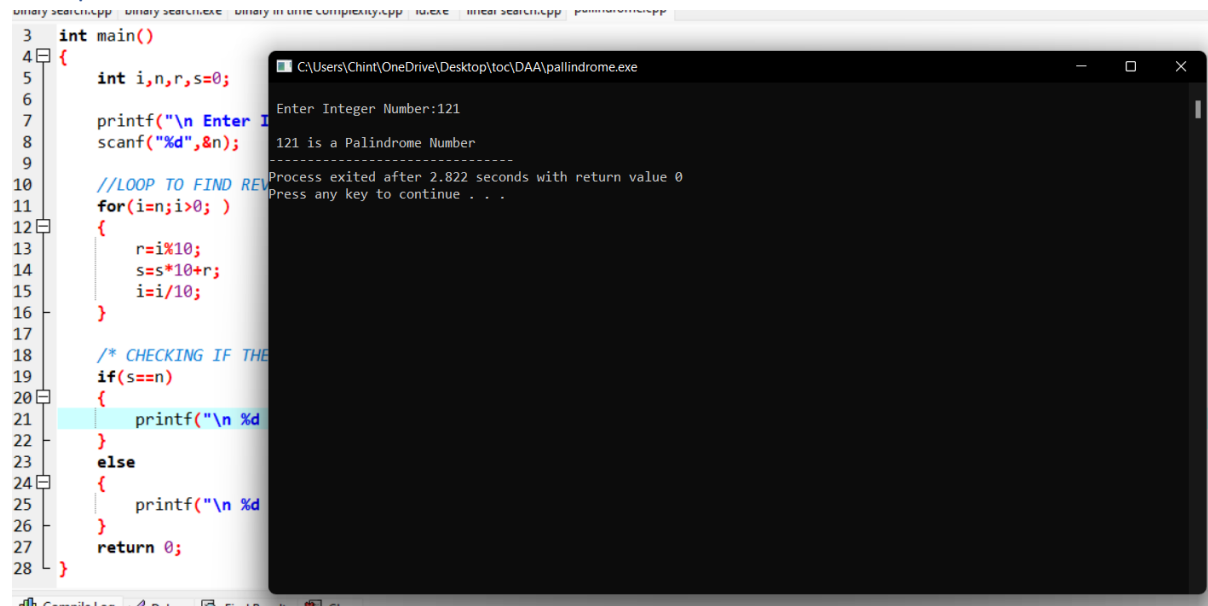
```
    printf("\n %d is not a Palindrome Number",n);
```

```
}
```

```
return 0;
```

```
}
```

Output:



```
3 int main()
4 {
5     int i,n,r,s=0;
6
7     printf("\n Enter Integer Number:");
8     scanf("%d",&n);
9
10    //LOOP TO FIND REVERSE
11    for(i=n;i>0; )
12    {
13        r=i%10;
14        s=s*10+r;
15        i=i/10;
16    }
17
18    /* CHECKING IF THE NUMBER IS PALINDROME */
19    if(s==n)
20    {
21        printf("\n %d is a Palindrome Number",n);
22    }
23    else
24    {
25        printf("\n %d is not a Palindrome Number",n);
26    }
27    return 0;
28 }
```

Enter Integer Number:121
121 is a Palindrome Number

Process exited after 2.822 seconds with return value 0
Press any key to continue . . .

14.Bubble sort:

Program;

```
#include<stdio.h>
```

```
int main(){
```

```
    int ele,count=0;
```

```
    printf("Enter total element: ");
```

```
    scanf("%d",&ele);
```

```
    int arr[ele];
```

```
    printf("Enter the elements: ");
```

```
    for (int i = 0; i < ele; i++){
```

```
        count++;
```

```
        scanf("%d",&arr[i]);
```

```
    }count++;
```

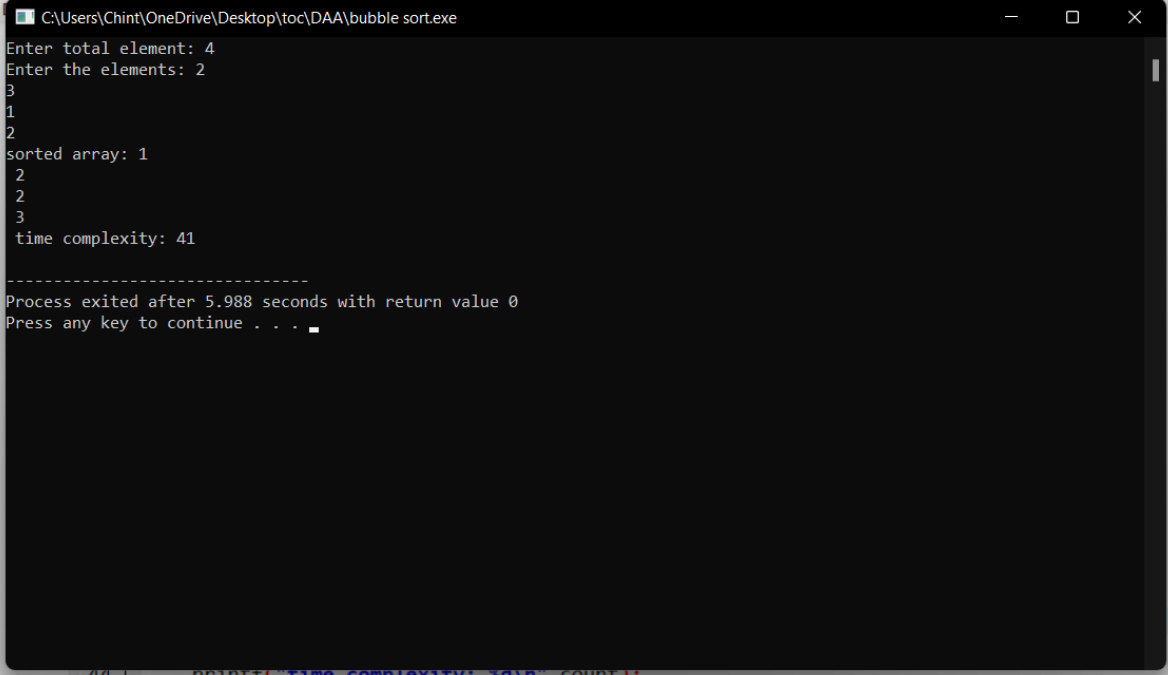
```
    for (int i = 0; i < ele; i++)
```

```
{
count++;
    for (int j =i+1; j < ele; j++)
    {
count++;
        if (arr[i]>arr[j])
        {
            count++;
            int temp=arr[i];
            count++;
            arr[i]=arr[j];
            count++;
            arr[j]=temp;
            count++;
        }
    }count++;

}count++;

printf("sorted array: ");
for (int i = 0; i < ele; i++)
{count++;
    count++;
    printf("%d \n ",arr[i]);
}count++;
printf("time complexity: %d\n",count);
}
```


Output:



```
C:\Users\Chint\OneDrive\Desktop\toc\DAA\bubble sort.exe
Enter total element: 4
Enter the elements: 2
3
1
2
sorted array: 1
2
2
3
time complexity: 41
-----
Process exited after 5.988 seconds with return value 0
Press any key to continue . . .
```

15.Reverse string:

Program:

```
#include<stdio.h>
```

```
int main(){
```

```
    char val[25];
```

```
    printf("enter the value: ");
```

```
    scanf("%s",&val);
```

```
    int count=0,c=0;
```

```
    while (val[count]!='\0'){
```

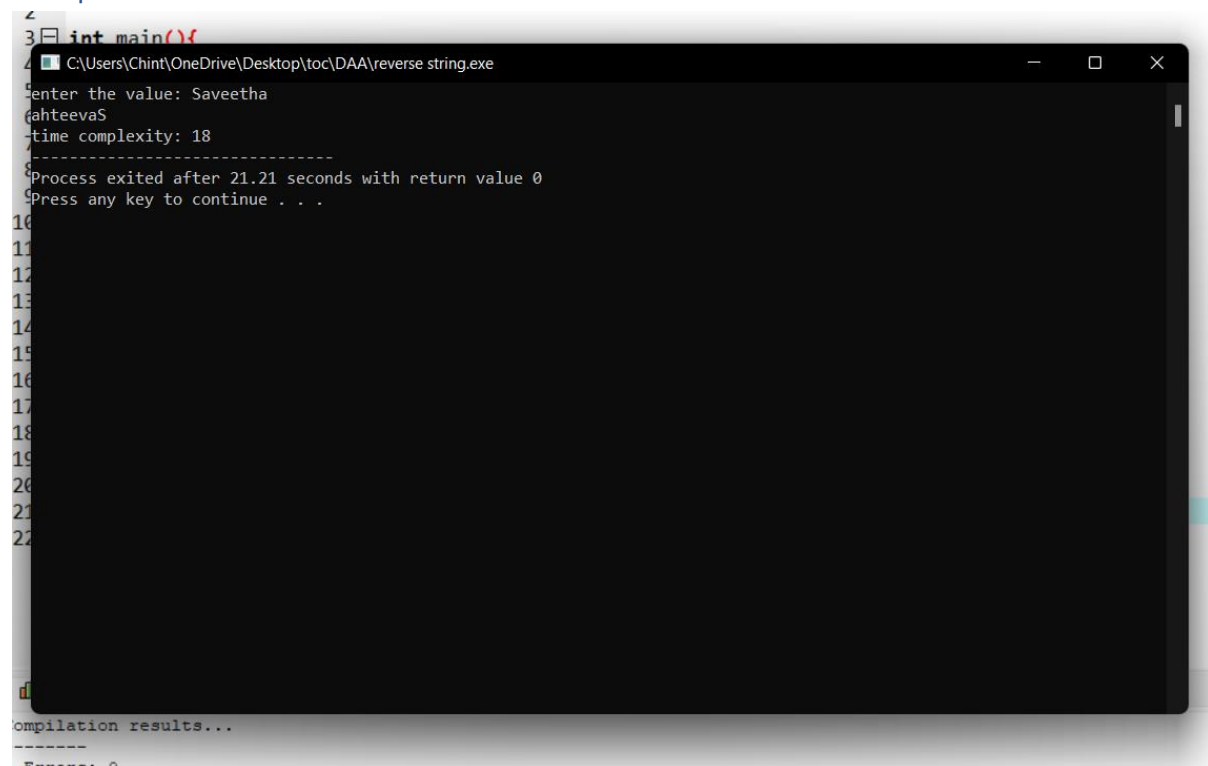
```
        count++;
```

```
        c++;
```

```
    }c++;
```

```
for(int i=count-1;i>=0;i--){  
    c++;  
    printf("%c",val[i]);  
}c++;  
printf("\ntime complexity: %d",c);  
}
```

Output:



```
3 int main(){  
4 C:\Users\Chint\OneDrive\Desktop\toC\DAA\reverse string.exe  
5 Enter the value: Saveetha  
6 SaehteevaS  
7 time complexity: 18  
8 -----  
9 Process exited after 21.21 seconds with return value 0  
10 Press any key to continue . . .  
11  
12  
13  
14  
15  
16  
17  
18  
19  
20  
21  
22  
Compilation results...  
-----  
Errors: 0
```

16.substring:

Program:

```
#include<stdio.h>
```

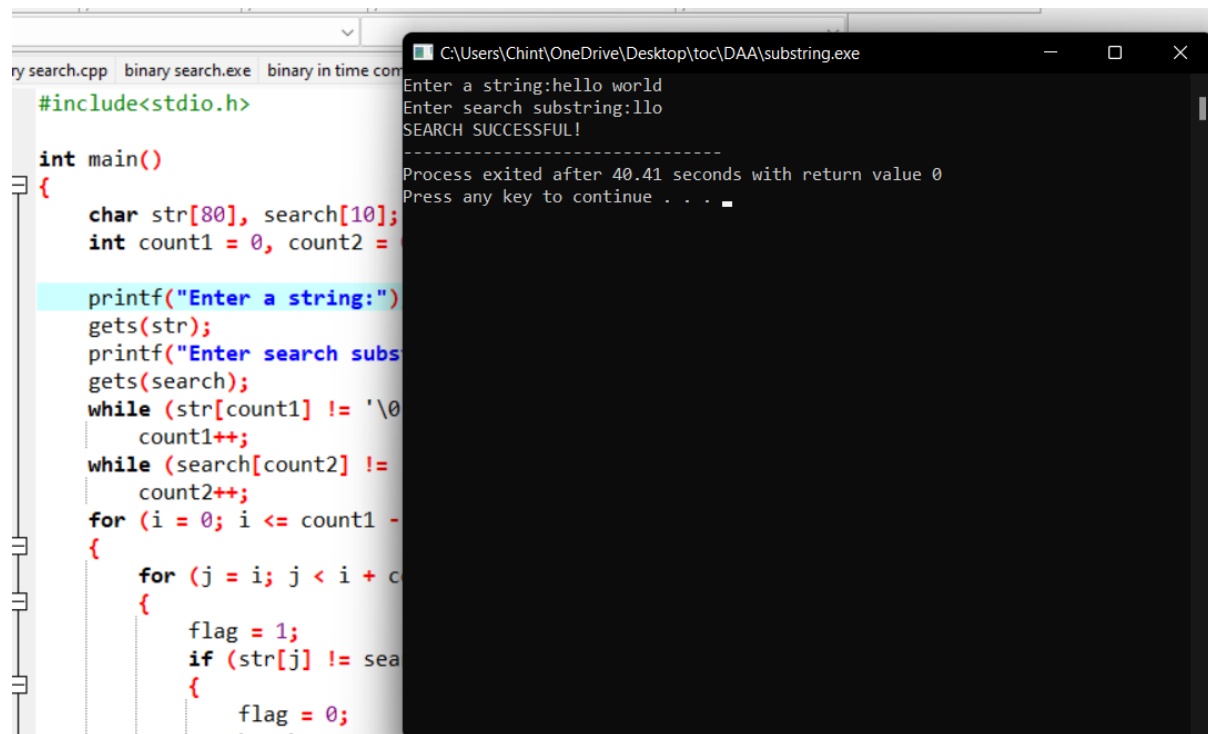
```
int main()  
{  
    char str[80], search[10];
```

```
int count1 = 0, count2 = 0, i, j, flag;

printf("Enter a string:");
gets(str);
printf("Enter search substring:");
gets(search);
while (str[count1] != '\0')
    count1++;
while (search[count2] != '\0')
    count2++;
for (i = 0; i <= count1 - count2; i++)
{
    for (j = i; j < i + count2; j++)
    {
        flag = 1;
        if (str[j] != search[j - i])
        {
            flag = 0;
            break;
        }
    }
    if (flag == 1)
        break;
}
if (flag == 1)
    printf("SEARCH SUCCESSFUL!");
else
    printf("SEARCH UNSUCCESSFUL!");

return 0;
}
```

Output:



The image shows a C++ IDE with a file named 'binary search.cpp' open. The code implements a substring search algorithm. It prompts the user to enter a string and a search substring. In the output window, the user has entered 'hello world' for the string and 'llo' for the search substring. The program successfully finds the substring and prints 'SEARCH SUCCESSFUL!'. Below this, it shows the process exit message: 'Process exited after 40.41 seconds with return value 0' and 'Press any key to continue . . .'. The code in the IDE is as follows:

```
#include<stdio.h>

int main()
{
    char str[80], search[10];
    int count1 = 0, count2 = 0;

    printf("Enter a string:");
    gets(str);
    printf("Enter search substring:");
    gets(search);
    while (str[count1] != '\0')
        count1++;
    while (search[count2] != '\0')
        count2++;
    for (i = 0; i <= count1 - count2; i++)
    {
        for (j = i; j < i + count2; j++)
        {
            if (str[j] != search[j - i])
            {
                flag = 0;
                break;
            }
        }
    }
}
```