

1) Sum of two numbers

Code :

```
#include<stdio.h>
#include<conio.h>
void main()
{
    int A, B, sum;
    scanf("%d %d",&A,&B);
    sum=A+B;
    printf("%d",&sum);
}
```

Algorithm

Step-1 Start

Step-2 Input first number say A O(1)

Step-3 Input second number say B O(1)

Step-4 SUM = A + B O(1)

Step-5 Display SUM O(1)

Step-6 Stop

Observations

Time complexity = $O(1) + O(1) + O(1) + O(1) = O(1)$. Its time complexity is constant in time.

2) Finding Celsius to Fahrenheit and Fahrenheit to Celsius

Code :

```
#include<stdio.h>
#include<conio.h>
void main()
```

```

{
    int c,f,C,F;
    printf("Enter the temperatures in Celsius and Fahrenheit
respectively \n");
    scanf("%d %d",&c,&f);
    F=((9/5)*c)+32;
    C=(5/9)*(f-32);
    printf("The converted temperatures in Fahrenheit and Celsius
respectively are %d and %d",&F,&C);
}

```

Algorithm

Step-1.	Start	
Step-2	Input temperature in Celsius say C,F	O(1)
Step-3	$F = (9.0/5.0 \times C) + 32$	O(1)
Step -4	$C = 5.0/9.0 (F - 32)$	O(1)
Step-5	Display Temperature in Fahrenheit F,C	O(1)
Step-6	Stop	

Observations

Time complexity = $O(1) + O(1) + O(1) + O(1) = O(1)$. Time complexity is constant time.

3) Area and Perimeter of square

Code :

```

#include<stdio.h>
#include<conio.h>
void main()
{
    int l,P,A;

```

```

printf("Enter side length of square \n");
scanf("%d",&l);
P=4*l;
A=l*l;

printf("The Perimeter and Area of the square is %d and %d
respectively",&P,&A);
}

```

Algorithm

Step-1	Start	
Step-2	Input Side Length of Square say l	O(1)
Step-3	Area = l x l	O(1)
Step-4	PERIMETER = 4 x l	O(1)
Step-5	Display AREA, PERIMETER	O(1)
Step-6	Stop	

Observations

Time complexity = $O(1)+O(1)+O(1)+O(1) = O(1)$. Time complexity is constant time.

4) Finding Compound interest

Code :

```

#include<stdio.h >
#include<conio.h>
void main()
{
    int P, r, n, CI;
    printf("Enter principle amount, interest rate(percentage), time
taken respectively");
    scanf("%d %d %d", &P, &r, &n);
}

```

```

    CI=P((1+(r/100))^n)-P;
    printf("The counpound interest is %d", &CI);
}

```

Algorithm

Step-1 Start

Step-2 Input value of P, N, R C O(1)

Step-3 $CI = P(1+R/100)^N - P$ O(1)

Step-4 Display CI O(1)

Step-5 Stop

Observations

Time complexity = $O(1) + O(1) + O(1) = O(1)$. Time complexity is constant time.

5) Swap two numbers using temporary variable

Code :

```

#include<stdio.h >
#include<conio.h>
void main()
{
    int a,b,t;
    printf("Enter two numbers a and b respectively to swap ");
    scanf("%d %d", &a, &b);
    printf("a=%d b=%d", &a, &b);
    temp=a;
    a=b;
    b=temp;
    printf("After swaping a=%d b=%d", &a, &b);
}

```

Algorithm

Step-1	Start	
Step-2	Input Two Numbers Say NUM1, NUM2	O(1)
Step-3	Display Before Swap Values NUM1, NUM2	O(1)
Step-4	TEMP = NUM1	O(1)
Step-5	NUM1 = NUM2	O(1)
Step-6	NUM2 = NUM1	O(1)
Step-7	Display After Swap Values NUM1, NUM2	O(1)
Step-8	Stop	

Observations

Time complexity = $O(1) + O(1) + O(1) + O(1) + O(1) + O(1) = O(1)$. Time complexity is constant time.

6) Finding smallest of two numbers

Code :

```
#include<stdio.h >
#include<conio.h>
void main()
{
    int n1, n2;
    printf("Enter any two numbers to compare ");
    scanf("%d %d ", &n1, &n2);
    if(n1<n2)
        printf("Smallest of the two is %d",&n1);
    else
        printf("Smallest of the two is %d",&n2);
}
```

Algorithm

Step-1	Start	
Step-2	Input two numbers say NUM1,NUM2	O(1)
Step-3	IF NUM1 < NUM2	
	THEN	
	print smallest is NUM1	O(1)
	ELSE	
	print smallest is NUM2	O(1)
	ENDIF	
Step-4	Stop	

Observations

Time complexity = $O(1) + \max(O(1), O(1)) = O(1) + O(1) = O(1)$. Time complexity is constant time. Time complexity is the worst time taken of an algorithm so we take the path which takes the longest time when an if condition is used.

7) Largest of three numbers

Code :

```
#include<stdio.h >
#include<conio.h>
void main()
{
    int a,b,c;
    printf("Print any three numbers to compare");
    scanf("%d %d %d", &a, &b, &c);
    if(a>b)
    {
        if(a>c)
            printf("%d is the largest", &a);
```

```

        else
            printf("%d is the largest",&c);
    }
    else
    {
        if(b>c)
            printf("%d is the largest", &b);
        else
            printf("%d is the largest", &c);
    }
}

```

Algorithm

Step-1	Start	
Step-2	Read three numbers say num1,num2, num3	O(1)
Step-3	if num1>num2 then go to step-5	
Step-4	If num2>num3	
	THEN	
	print num2 is largest	O(1)
	ELSE	
	print num3 is largest	O(1)
	END IF	
GO TO	Step-6	
Step-5	IF num1>num3	
	THEN	
	print num1 is largest	O(1)
	ELSE	
	print num3 is largest	O(1)
	ENDIF	
Step-6	Stop	

Observations

Time complexity = $O(1) + \max(O(1) + O(1), O(1) + O(1)) = O(1) + O(1) + O(1) = O(1)$. Time complexity is constant time. Time complexity is the worst time taken of an algorithm so we take the path which takes the longest time when an if condition is used.

8) Even numbers between 1 to 50

Code :

```
#include<stdio.h >
#include<conio.h>
void main()
{
    int i;
    for(i=1,i<=50,i++)
    {
        if((i%2)==0)
            printf("%d \n", &i);
    }
}
```

Algorithm

Step-1 Start

Step-2 $I = 1$ $O(1)$

Step-3 IF ($I > 50$) THEN
 GO TO Step-7
 ENDIF

Step-4 IF (($I \% 2$) =0) THEN $25 * O(1)$
 Display I
 ENDIF

Step-5	$I = I + 1$	$50 * O(1)$
Step-6	GO TO Step--3	
Step-7	Stop	

Observations

Time complexity = $O(1) + 25 * O(1) + 50 * O(1) = O(1)$. Time complexity is constant time because its time doesn't depend on any variable.

9) Sum of series 1+2+3+...+n

Code :

```
#include<stdio.h >
#include<conio.h>
void main()
{
    int n, i, sum=0;
    printf("Enter the value of n \n");
    scanf("%d", &n);
    for(i=1,i<=n,i++)
    {
        sum=sum+i;
    }
    printf("The sum of the series is %d", &sum);
}
```

Algorithm

Step-1	Start	
Step-2	Input Value of N	$O(1)$
Step-3	$I = 1, SUM = 0$	$O(1)$
Step-4	IF ($I > N$) THEN GO TO 8 Step-8	ENDIF
Step-5	$SUM = SUM + I$	

Step-6	$I = I + 1$	$O(n)$
Step-7	Go to step-4	
Step-8	Display value of SUM	$O(1)$
Step-9	Stop	

Observations

Time complexity = $O(1) + O(1) + O(n) + O(1) = O(n)$. Time complexity is linear time here because the time complexity depends on the value of the number of times the loop repeats which is 'n' in this case.