



# WEST BENGAL STATE UNIVERSITY

B. Sc. Honours Semester–I CBCS Examination 2022

**Programming Fundamentals using C/C++ Lab Practical**

Laboratory Note Book

SUBJECT : COMPUTER SCIENCE

PAPER CODE : CSMACOR01P

Reg no.

Roll.

No.

## Index

Sl. No.	Description	Date	Signature
1	WAP to print the sum and product of digits of an integer.		
2	WAP to reverse a number.		
3	WAP to compute the sum of the first n terms of the following series : $S = 1 + 1/2 + 1/3 + 1/4 + \dots$		
4	WAP to compute the sum of the first n terms of the following series : $S = 1 - 2 + 3 - 4 + \dots$		
5	WAP to check if palindrome (By taking user-input)		
6	Write a function to find whether a given no. is prime or not. Use the same to generate the prime numbers less than 100.		
7	WAP to compute the factors of a given number.		
8	WAP to print a triangle (By taking user-input)		
9	WAP to calculate GCD of two numbers (i) with recursion (ii) without recursion.		
10	Create a class object with show as virtual function. Inherit two other classes sphere and cube from object. Calculate volume of sphere and cube using necessary parameters and member function.		
11	Copy the contents of one text file to another file, after removing all whitespaces.		
12	Design a class polar which describes a point in the plain using polar coordinates radius and angle. Use overloaded operator to add two objects of Polar.		

## **PROGRAM 1: WAP to print the sum and product of digits of an integer.**

### **Algorithm :**

STEP\_1: Start  
STEP\_2: sum := 0, product := 1  
STEP\_3: Read integer i  
STEP\_4: while ( i != 0 )  
STEP\_4.1: sum  $\leftarrow$  sum + ( i % 10 )  
STEP\_4.2: product  $\leftarrow$  product \* ( i % 10 )  
STEP\_4.3: i  $\leftarrow$  i / 10  
STEP\_5: end while  
STEP\_6: Print sum and product  
STEP\_7: End

### **Program Code :**

```
#include<iostream>
using namespace std;
int main(void){
    int sum = 0, product = 1, i{};
    std::cout << "Enter an Integer : " && std::cin >> i;
    while (i != 0) {
        sum += (i % 10);
        product *= (i % 10);
        i /= 10;
    }
    std::cout << "The sum is : " << sum << std::endl;
    std::cout << "The product is : " << product << std::endl;
    return 0;
}
```

### **Output :**

#### **SET-1 :**

Enter an Integer : 1010010  
The sum is : 3  
The product is : 0

#### **SET-2 :**

Enter an Integer : 119  
The sum is : 11  
The product is : 9

### **Discussion :**

Time Complexity:  $O(n)$

Space Complexity:  $O(1)$

Program Limitation: Integer input beyond the range of 0 to INT\_MAX will return inaccurate results.

## PROGRAM 2: WAP to reverse a number

### Algorithm :

```
STEP_1: Start
STEP_2: reverse := 0
STEP_3: Read integer i
STEP_4: while ( i != 0 )
STEP_4.1: reverse  $\leftarrow$  ( reverse * 10 ) + ( i % 10 )
STEP_4.2: i  $\leftarrow$  i / 10
STEP_5: end while
STEP_6: i := reverse
STEP_7: Print i
STEP_8: End
```

### Program Code :

```
#include <iostream>
using namespace std;
int main(void){
    int i{}, reverse = 0;
    std::cout << "Enter an Integer : " && std::cin >> i;
    while (i != 0){
        reverse = (reverse * 10) + (i % 10);
        i /= 10;
    }
    i = reverse;
    std::cout << "Reverse of the given integer is : " << i << endl;
    return 0;
}
```

### Output :

*SET-1 :*

Enter an Integer : 17625

Reverse of the given integer is : 52671

*SET-2 :*

Enter an Integer : -1661

Reverse of the given integer is : -1661

### Discussion :

Time Complexity:  $O(n)$

Space Complexity:  $O(1)$

Program Limitation: Integer input beyond the range of INT\_MIN to INT\_MAX will return inaccurate results.

**PROGRAM 3: WAP to compute the sum of the first n terms of the following series :  $S = 1 + 1/2 + 1/3 + 1/4 + \dots$**

**Algorithm :**

```
STEP_1: Start
STEP_2: S := 0
STEP_3: Read integer i
STEP_4: while ( i != 0 )
STEP_4.1: S  $\leftarrow$  S + (1/i)
STEP_4.2: i  $\leftarrow$  i - 1
STEP_5: end while
STEP_6: Print S
STEP_7: End
```

**Program Code :**

```
#include <iostream>
using namespace std;
int main() {
    int i{};
    double S = 0;
    std::cout << "Enter the value of n : " && std::cin >> i;
    if (i < 1) return -1;
    while ( i != 0 ) {
        S += ( 1 / double(i));
        --i;
    }
    std::cout << fixed << setprecision(2) << "The required sum is : " << S << std::endl;
    return 0;
}
```

**Output :**

**SET-1 :**

```
Enter the value of n : 42
The required sum is : 4.33
```

**SET-2 :**

```
Enter the value of n : 2
The required sum is : 1.5
```

**Discussion :**

Time Complexity:  $O(n)$

Space Complexity:  $O(1)$

Program Limitation: Input number is a Natural Number and beyond the range of 1 to INT\_MAX will return inaccurate results.

**PROGRAM 4: WAP to compute the sum of the first n terms of the following series :  $S = 1 - 2 + 3 - 4 + \dots$**

**Algorithm :**

```
STEP_1: Start
STEP_2: S := 0
STEP_3: Read integer i
STEP_4: if ( i % 2 == 0)
STEP_4.1: S ← S - (i / 2)
STEP_5: else
STEP_5.1: S ← (i + 1)/2
STEP_6: Print S
STEP_7: End
```

**Program Code :**

```
#include <iostream>
using namespace std;
int main(){
    int i {}, S = 0;
    std::cout << "Enter the value of n : " && std::cin >> i;
    if (i < 1) return -1;
    S = ( i % 2 == 0) ? S - (i / 2) : ( i + 1) / 2;
    std::cout << "The obtained result is : " << S << endl;
    return 0;
}
```

**Output :**

***SET-1 :***

Enter the value of n : 16  
The obtained result is : -8

***SET-2 :***

Enter the value of n : 47  
The obtained result is : 24

**Discussion :**

Time Complexity:  $O(1)$

Space Complexity:  $O(1)$

Program Limitation: Input number is a Natural Number and beyond the range of 1 to `INT_MAX` will return inaccurate results.

**PROGRAM 5: Write a function that checks whether a given string is Palindrome or not. Use this function to find whether the string entered by user is Palindrome or not.**

**Algorithm :**

```
STEP_1: Start
STEP_2: function check_if_palindrome
STEP_2.1: Input string
STEP_2.2: n := string.length
STEP_2.3: for (i := 0; i < n / 2; ++i)
STEP_2.3.1: if (string[i] != string [n - i - 1]) return false
STEP_2.4: end for
STEP_2.5: return true
STEP_3: end function check_if_palindrome
STEP_4: Read string str
STEP_5: if ( check_if_palindrome(str) )
STEP_5.1: Print "Is a Palindrome"
STEP_6: else
STEP_6.1.: Print "Not a Palindrome"
STEP_7: End
```

**Program Code :**

```
#include <iostream>
using namespace std;
bool check_if_palindrome(string str) {
for (size_t i = 0; i < str.length() / 2; ++i)
if (str[i] != str[str.length() - i - 1]) return false;
return true;
}
int main(void){
string str{};
std::cout << "Enter a string : " && std::cin >> str;
std::cout << (check_if_palindrome(str) ? "Is a palindrome" : "Not a palindrome") << endl;
}
```

**Output :**

**SET-1 :**

```
Enter a string : madam
Is a palindrome
```

**SET-2 :**

```
Enter a string : me
Not a palindrome
```

**Discussion :**

Time Complexity: O(n)

Space Complexity: O(1)

Program Limitation: Can check only one word at a time. The input of the program is case sensitive.

**PROGRAM 6: Write a function to find whether a given no. is prime or not. Use the same to generate the prime numbers less than 100**

**Algorithm :**

```
STEP_1: Start
STEP_2: function is_prime
STEP_2.1: Input x
STEP_2.2: for (i := 2; i <= x/2; ++i)
STEP_2.2.1: if (x % i == 0)
STEP_2.2.1.1: is_prime := false
STEP_2.2.1.2: break
STEP_2.2.2: else
STEP_2.2.2.1: is_prime := true
STEP_2.3: end for
STEP_2.4: if (is_prime)
STEP_2.4.1: Print "Is a prime number"
STEP_2.5: else
STEP_2.5.1: Print "Not a prime number"
STEP_2.6: return is_prime
STEP_3: end function is_prime
STEP_3: function prime_under_100
STEP_3.1: for (i := 0; i < 100; ++i)
STEP_3.1.1: if (is_prime(i))
STEP_3.1.1.1: Print i
STEP_3.2: end for
STEP_4: end function prime_under_100
STEP_5: Read i
STEP_6: print a menu to choose whether to check if variable i is a prime number or not else
choose to print all prime integers under 100
STEP_7: perform selected operation
STEP_8: End
```

**Program Code :**

```
#include <iostream>
using namespace std;
bool is_prime(unsigned x){
    bool is_prime{};
    for (unsigned i = 2; i <= x/2 ; i++) {
        if (x % i == 0){is_prime = false;break;}
        is_prime = true;}
    return is_prime;}
void prime_under_100(){ // function for printing prime numbers under 100
    std::cout << "Prime Numbers under 100 : ";
    for (unsigned i = 0; i < 100; i++)
        std::cout << (is_prime(i) ? to_string(i) : "" ) << (is_prime(i) ? " " : "" );
    std::cout << endl;
}
```



```

int main(void){
    unsigned i{};
    std::cout << "1. check prime or not " << std::endl;
    std::cout << "2. print prime numbers under 100 " << std::endl;
    std::cout << "Enter your choice : " && std::cin >> i;
    switch (i) {
        case 1: {
            unsigned x{};
            std::cout << "Enter a positive integer : ";
            std::cin >> x;
            std::cout << ((is_prime(x)) ? "Is a prime number" : "Not a prime number") << endl;
            break;
        }
        case 2: prime_under_100();break;
        default: std::cout << "Invalid choice" << std::endl; return -1;
    }
    return 0;
}

```

## Output :

### **SET-1 :**

```

1. check prime or not
2. print prime numbers under 100
Enter your choice : 1
Enter a positive integer : 100
Not a prime number

```

### **SET-2 :**

```

1. check prime or not
2. print prime numbers under 100
Enter your choice : 2
Prime Numbers under 100 : 3 5 7 11 13 17 19 23 29 31 37 41 43 47 53 59 61 67 71 73 79 83
89 97

```

## Discussion :

Time Complexity:  $O(n)$

Space Complexity:  $O(1)$

Program Limitation: Inside the function of checking for prime numbers, input number should be a Natural Number and beyond the range of 1 to INT\_MAX will return inaccurate results.

## PROGRAM 7: WAP to compute the factors of a given number

### Algorithm :

```
STEP_1: Start
STEP_2: Read integer i
STEP_3: if (i == 0)
STEP_3.1: exit program
STEP_4: else if (i < 0)
STEP_4.1: i ← i / -1
STEP_4.2: Print -1
STEP_5: else
STEP_5.1: Print 1
STEP_6: while (i != 1)
STEP_6.1: static factor := 2
STEP_6.2: if (i % factor == 0)
STEP_6.2.1: print factor
STEP_6.2.2: i ← i / factor
STEP_6.3: else
STEP_6.3.1: f ← f + 1
STEP_7: end while
STEP_8: End
```

### Program Code :

```
#include <iostream>
using namespace std;
int main(void){
    int n{};
    std::cout << "Enter an Integer : " && std::cin >> n;
    if (n == 0) exit(EXIT_SUCCESS);
    std::cout << "Factors of " << n << " is : ";
    if (n > 0){std::cout << " 1 ";}
    if (n < 0){std::cout << "-1 "; n /= -1;}
    while ( n != 1 ){
        static int f = 2;
        if (n % f == 0) {
            std::cout << f << " ";
            n /= f;
            continue;
        }
        ++f;
    }
    std::cout << std::endl;
}
```

### Output :

#### SET-1 :

```
Enter an Integer : -1616
Factors of -1616 is : -1 2 2 2 2 101
```

**SET-2 :**

Enter an Integer : 126  
 Factors of 126 is : 1 2 3 3 7

**Discussion :**

Time Complexity:  $O(n)$   
 Space Complexity:  $O(1)$   
 Program Limitation: Integer input beyond the range of INT\_MIN to INT\_MAX will return inaccurate results.

**PROGRAM 8: WAP to print a triangle of stars as follows (take number of lines from user) :**

```

      *
    * * *
  * * * * *
* * * * * * *
```

**Algorithm :**

```

STEP_1: Start
STEP_2: Read positive integer lines
STEP_3: for (i := 0; i < lines; ++i)
STEP_3.1: for (j := 0; j < lines; ++j)
STEP_3.1.1: if (i + j >= lines)
STEP_3.1.1.1: print "*"
STEP_3.1.2: else
STEP_3.1.2.1: print " "
STEP_3.2: end for
STEP_3.3: for (j := 0; j < i; ++j)
STEP_3.3.1: Print "*"
STEP_3.4: end for
STEP_3.4: endl
STEP_4: end for
STEP_5: End
```

**Program Code :**

```

#include <iostream>
using namespace std;
int main(void){
  int lines{};
  std::cout << "Enter the height of the triangle : " && std::cin >> lines;
  for (int i = 0; i < lines; i++){
    for (int j = 0; j <= lines; j++){std::cout << (i + j >= lines ? "*" : " ");}
    for (int j = 0; j < i; j++){std::cout << "* ";}
    std::cout << endl;
  }
}
```

## Output :

### SET-1 :

Enter the height of the triangle : 4

```
  *
 * * *
 * * * * *
 * * * * * * *
```

### SET-2 :

Enter the height of the triangle : 2

```
  *
 * * *
```

## Discussion :

Time Complexity:  $O(n)$

Space Complexity:  $O(1)$

Program Limitation: Integer input beyond the range of INT\_MIN to INT\_MAX will return inaccurate results.

## Program 9 : WAP to calculate GCD of two numbers (i) with recursion (ii) without recursion.

### Algorithm :

#### (i) Using recursion

```
STEP_1: Start
STEP_2: function GCD
STEP_2.1: input positive integer n1 and n2
STEP_2.2: if ( n1 == 0)
STEP_2.2.1: return n2
STEP_2.3: else
STEP_2.3.1: return GCD ( n2 % n1, n1)
STEP_3: end function GCD
STEP_4: Read positive integer n1 and n2
STEP_5: call GCD function with n1 and n2 as parameters
STEP_6: Print the value returned by the function
STEP_7: End
```

#### (ii) Without using recursion

```
STEP_1: Start
STEP_2: Read positive integer n1 and n2
STEP_3: if ( n1 < 1 or n2 < 1)
    print "Invalid Input" and exit Program
STEP_4: Set the minimum value between n1 and n2 inside variable named "min"
STEP_5: While ( min != 1)
STEP_5.1: if ( n1 % min == 0 and n2 % min == 0)
STEP_5.1.2: print min
STEP_5.2: end if
STEP_5.3: --min
STEP_6: end While
STEP_7: End
```

## Program Code :

### (i) Using recursion

```
#include <iostream>
#include <algorithm>
unsigned GCD (unsigned n1, unsigned n2) {
    if (n1 == 0) return n2;
    return GCD(n2 % n1 , n1);
}
int main(){
    unsigned n1 = 0, n2 = 0;
    std::cout << "Enter two positive integers : " && std::cin >> n1 >> n2;
    if ( n1 < 1 || n2 < 1) {
        std::cout << "invalid input" << std::endl;
        exit(EXIT_FAILURE);
    }
    std::cout << "The GCD of " << n1 << " and " << n2 << " is : " << GCD(n1, n2) << std::endl;
    return 0;
}
```

### (ii) Without using recursion

```
#include <iostream>
#include <algorithm>
int main(){
    int n1 = 0, n2 = 0;
    std::cout << "Enter two positive integers : " && std::cin >> n1 >> n2;
    if ( n1 < 1 || n2 < 1) {
        std::cout << "invalid input" << std::endl;
        exit(EXIT_FAILURE);
    }
    int min = std::min(n1, n2);
    while (min != 1) {
        if (n1 % min == 0 && n2 % min == 0) break;
        --min;
    }
    std::cout << "The GCD of " << n1 << " and " << n2 << " is : " << min << std::endl;
    return 0;
}
```

## Output :

### SET-1 :

Enter two positive integers : 72 63  
The GCD of 72 and 63 is : 9

### SET-2 :

Enter two positive integers : 32 48  
The GCD of 32 and 48 is : 16

## Discussion :

Time Complexity:  $O(n)$

Space Complexity: (i) Using recursion :  $O(n)$  (ii) Without using recursion :  $O(1)$

Program Limitation: Input number is a Natural Number and beyond the range of 1 to `INT_MAX` will return inaccurate results.

**PROGRAM 10 : Create a class object with show as virtual function. Inherit two other classes sphere and cube from object. Calculate volume of sphere and cube using necessary parameters and member function.**

## Algorithm :

STEP\_1: Start

STEP\_2: Declare class “object” with data member “volume” and virtual member function “show” that prints the value of the data member volume

STEP\_3: Declare a derived class “sphere” that inherits the base class “object” with data member “radius” and a parameterized constructor that takes an integer input “radius” and sets “radius” and calculates and sets “volume” of the object

STEP\_4: Declare a derived class “cube” that inherits the base class “object” with data member “side” and a parameterized constructor that takes an integer input “side” and sets “side” and calculates and sets “volume” of the object

STEP\_5: Declare pointer object \*obj;

STEP\_6: Choose between either “sphere” or “cube” derived class to create an instance

STEP\_7: Write the address of the created instance to the created pointer variable “obj” of class “object”

STEP\_8: Read input

STEP\_9: Print volume by calling the “show” function

STEP\_10: End

## Program Code :

```
#include <iostream>
#include <cmath>
#define PI 3.14
using namespace std;
class object{
public:
double volume{};
virtual void show(){std::cout << "Volume is : " << volume << std::endl;};
};
class sphere: public object{
double radius{};
public:
sphere(double radius){
this->radius = radius;
object::volume = 4/3 * PI * pow(radius, 3);
};
~sphere(){};
};
```

```

class cube: public object{
double side{};
public:
cube(double side){this->side = side;object::volume = pow(side, 3);
}
~cube(){};
};
int main(void){
double n{};
int choice{};
object *obj;
std::cout << "Object type : (1) Sphere (2) Cube" << std::endl;
std::cout << "Enter choice : " && std::cin >> choice;
switch(choice) {
case 1 :{
std::cout << "Enter the radius the sphere : " && std::cin >> n;
if(n < 0){std::cout << "Invalid input!!" << std::endl; exit(EXIT_FAILURE);};
sphere obj1(n);
obj = &obj1;
break;}
case 2 :{
std::cout << "Enter the length of each side of the cube : " && std::cin >> n;
if(n < 0){std::cout << "Invalid input!!" << std::endl; exit(EXIT_FAILURE);};
cube obj1(n);
obj = &obj1;
break;}
default :{std::cout << "Invalid Choice!!" << std::endl;exit(EXIT_FAILURE);}
}
obj->show();
return 0;
}

```

## Output :

### Set-1 :

Object type : (1) Sphere (2) Cube  
Enter choice : 1  
Enter the radius the sphere : 21  
Volume is : 29079.5

### Set-2 :

Object type : (1) Sphere (2) Cube  
Enter choice : 2  
Enter the length of each side of the cube : 4.5  
Volume is : 91.125

## Discussion :

Time Complexity :  $O(n)$   
Space Complexity :  $O(1)$   
Program limitations : In the above program, input range of “side” and “range” is within range of 1 to INT\_MAX. Input beyond this limit may return inaccurate results.

## **Program 11 : Copy the contents of one text file to another file, after removing all whitespaces.**

### **Algorithm :**

```
STEP_1: Start
STEP_2: function copyToAnotherFile
STEP_2.1: Input filename "fin"
STEP_2.2: Input filename "fout"
STEP_2.3: Declare two file pointers fl1 and fl2
STEP_2.4: Declare ch
STEP_2.5: Set fl1 file pointer at the beginning of "fin" in read mode
STEP_2.6: Set fl2 file pointer at the beginning of "fout" in write mode
STEP_2.7: Read character fl1 and set it inside ch
STEP_2.8: While ((ch = fgetc(fl1))!= EOF)
STEP_2.8.1: Print ch
STEP_2.8.2: if (ch != ' ')
STEP_2.8.2.1: Insert ch through fl2 pointer inside "fout" file
STEP_2.8.3: end if
STEP_2.9: end While
STEP_2.10: Close fl2 file pointer
STEP_2.11: Set fl2 file pointer at the beginning of "fout" in read mode
STEP_2.12: Read character fl2 and set it inside ch
STEP_2.13: While ((ch = fgetc(fl1))!= EOF)
STEP_2.13.1: Print ch
STEP_2.14: end While
STEP_3: end function copyToAnotherFile
STEP_4: Read input and output filenames
STEP_5: Call function copyToAnotherFile with both filenames as arguments
STEP_6 : End
```

### **Program Code :**

```
#include <iostream>
#include <fstream>
#include <stdlib.h>
#include <unistd.h>
using namespace std;
void copyToAnotherFile(char fin[], char fout[]){
FILE *fl1,*fl2;
char ch;
fl1 = fopen(fin, "r");
fl2 = fopen(fout, "w");
std::cout << "[CONTENTS OF \"" << fin << "\" FILE]" <<std::endl;
while ((ch = fgetc(fl1))!= EOF){
std::cout << ch;
if (ch != ' '){fputc(ch, fl2);};
}
}
```



```

std::cout << "[CONTENTS OF \"" << fout << "\" FILE]" << std::endl;
fclose(fl2);
fl2 = fopen(fout, "r");
while ((ch = fgetc(fl2)) != EOF) {std::cout << ch;}
fclose(fl1);
fclose(fl2);
}
int main(int argc, char **argv){
switch (argc) {
case 1 : {std::cout << "Specify Input and Output file" << std::endl;exit(EXIT_FAILURE);}
case 2 : {std::cout << "Specify Output file" << std::endl;exit(EXIT_FAILURE);}
case 3 : {if (access(argv[1], F_OK) != 0) {
std::cout << argv[1] << "Input file not found!!" << std::endl;
exit(EXIT_FAILURE);}
copyToAnotherFile(argv[1], argv[2]);
break;
}
default : {std::cout << "Too many arguments" << std::endl;exit(EXIT_FAILURE);}
}
return 0;
}

```

## Output :

**Set-1 :** [\$ ./a.out sample.txt new.txt ]  
[CONTENTS OF "sample.txt" FILE]  
foot ball  
water melon  
jack pot  
[CONTENTS OF "new.txt" FILE]  
football  
watermelon  
jackpot

**Set-2 :** [\$ ./a.out sample2.txt new2.txt ]  
[CONTENTS OF "sample2.txt" FILE]  
butter flies  
robin hood  
note book  
[CONTENTS OF "new2.txt" FILE]  
butterflies  
robinhood  
notebook

## Discussion :

Time Complexity:  $O(n)$

Space Complexity:  $O(1)$

Program Limitation: In the above program, only plain text files are allowed as input file.

**Program 12 : Design a class polar which describes a point in the plain using polar coordinates radius and angle. Use overloaded operator to add two objects of Polar.**

**Algorithm :**

STEP\_1: Start  
STEP\_2: Declare class “polar” with data members “radius” and “angle”, member functions “set\_data” to read and set “radius” and “angle”  
STEP\_3: Inside the class “polar”, “+” operator is overloaded in order to add two polar objects together by using the mathematical rules of addition of polar coordinates.  
STEP\_4: In the “main” function, create three polar objects “obj1”, “obj2” and “obj3”  
STEP\_5: Call “set\_data” member function over “obj1” and “obj2” and set the values  
STEP\_6:  $\text{obj3} \leftarrow \text{obj1} + \text{obj2}$   
STEP\_7: print “radius” and “angle” which were stored data inside “obj3”  
STEP\_8: End

**Program Code :**

```
#include <iostream>
#include <cmath>
#define PI 3.14
class polar{
double radius{}, angle{};
public:
void set_data(){
double r{}, a{};
std::cout << "Enter radius : " && std::cin >> r;
std::cout << "Enter angle (in degree[°]) : " && std::cin >> a;
radius = r;
angle = a;
}
double deg_to_rad(double angle){return PI / double(180) * angle;};
double rad_to_deg(double angle){return double(180) / PI * angle;};
polar operator + (polar obj){
polar temp_p;
double temp_1 {}, temp_2 {}, temp_rad {};
double obj_angle_in_rad = (deg_to_rad(obj.angle));
double temp_angle_in_rad = (deg_to_rad(angle));
temp_1 = (radius * cos(temp_angle_in_rad)) + (obj.radius * cos(obj_angle_in_rad));
temp_2 = (radius * sin(temp_angle_in_rad)) + (obj.radius * sin(obj_angle_in_rad));
temp_rad = atan(temp_2 / temp_1);
temp_p.radius = sqrt(temp_1 * temp_1 + temp_2 * temp_2);
temp_p.angle = rad_to_deg(temp_rad);
return temp_p;
}
void display(){std::cout << "Polar value : " << radius << " ∠ " << angle << "°" <<
std::endl;};
};
```

```

int main(void){
polar obj1, obj2, obj3;
obj1.set_data();
obj1.display();
obj2.set_data();
obj2.display();
obj3 = obj1 + obj2;
std::cout << "Value after addition" << std::endl;
obj3.display();
return 0;
}

```

## Output :

### Set-1 :

```

Enter radius : 6
Enter angle (in degree[°]) : 120
Polar value : 6 ∠ 120°
Enter radius : 10
Enter angle (in degree[°]) : 36.9
Polar value : 10 ∠ 36.9°
Value after addition
Polar value : 12.268 ∠ 65.9594°

```

### Set-2 :

```

Enter radius : 20
Enter angle (in degree[°]) : 22.5
Polar value : 20 ∠ 22.5°
Enter radius : 35
Enter angle (in degree[°]) : 75
Polar value : 35 ∠ 75°
Value after addition
Polar value : 49.7773 ∠ 56.4094°

```

## Discussion :

Time Complexity:  $O(n)$   
Space Complexity:  $O(1)$