BON SECOURS ARTS AND SCIENCE COLLEGE FOR WOMEN

(AFFILIATED TO MOTHER TERESA UNIVERSITY, KODAIKANAL)

MADURAI ROAD, BEGAMPUR, DINDIGUL – 02.

DEPARTMENT OF COMPUTER SCIENCE

RECORD NOTE BOOK

I SEMESTER

BSC COMPUTER SCIENCE

OOPS USING C++ LAB - U23CSP11

NAME : _____

REG. NO : _____

CLASS : _____

YEAR : _____

BON SECOURS ARTS AND SCIENCE COLLEGE FOR WOMEN.

(AFFILIATED TO MOTHER TERESA UNIVERSITY, KODAIKANAL)

MADURAI ROAD, BEGAMPUR, DINDIGUL – 02.

BONAFIDE CERTIFICATE

Registration Number										

Certified that this is the approved record of practical work done by _____ *of* _____*during the academic year* _____ *in the OOPS using C++ Lab(U23CSP11)*

Staff In-charge **Head of the Department**

Submitted for the University practical examination held on _____ .

Internal Examiner **External Examiner**

CONTENT

Ex. No: 1

GREATEST AMONG THREE NUMBERS.

Date:

AIM

To create a C++ program to find the greatest among three given numbers.

ALGORITHM

STEP1: Start the program.

STEP 2: Declare and include the needed variables & functions.

STEP 3: Define the Execution Method to find the greatest among three given numbers.

STEP 4: Save and compile the program

STEP 5: Run the program & Display the result.

STEP 6: Stop the program.

PROGRAM

```
#include <iostream>
#include<conio.h>
int main()
{
   double n1, n2, n3;
   cout << "Enter three numbers: ";
   cin >> n1 >> n2 >> n3;
   if(n1 >= n2 && n1 >= n3)
      cout << "Largest number: " << n1;
   else if(n2 >= n1 && n2 >= n3)
      cout << "Largest number: " << n2;
   else
      cout << "Largest number: " << n3;
    getch();
    return 0;
}
```

OUTPUT

```
Enter three numbers:  2.5     6     2.4
Largest number: 6
```

RESULT

Thus the program has been executed successfully.

Ex. No: 2

SWAPPING THE NMBERS

Date:

AIM

To create a C++ program to swap the given numbers.

ALGORITHM

STEP1: Start the program.

STEP 2: Declare and include the needed variables & functions.

STEP 3: Define the Execution Method to swap the given numbers using temp variable.

STEP 4: Save and compile the program

STEP 5: Run the program & Display the result.

STEP 6: Stop the program.

PROGRAM

```
#include <iostream>

#include<conio.h>

int main()
{
   int num1, num2, temp;
   cout<<"Enter 1st Number: ";
   cin>>num1;
   cout<<"Enter 2nd Number: ";
   cin>>num2;
   cout<<"Before Swapping: First Number: "<<num1<<" Second Number: "<<num2;
   temp=num1;
   num1=num2;
   num2=temp;
   cout<<"\nAfter Swapping: First Number: "<<num1<<" Second Number: "<<num2;
     getch();
     return 0;
}
```

OUTPUT

Enter 1st Number: 100

Enter 2nd Number: 50

Before Swapping: First Number: 100 Second Number: 50

After Swapping: First Number:50 Second Number: 100

RESULT

Thus the program has been executed successfully.

Ex. No: 3

SUM & AVERAGE OF GIVEN NUMBERS.

Date:

AIM

To create a C++ program to find the sum and average of given numbers.

ALGORITHM

STEP1: Start the program.

STEP 2: Declare and include the needed variables & functions.

STEP 3: Define the Execution Method for finding the sum & average of three numbers.

STEP 4: Save and compile the program

STEP 5: Run the program & Display the result.

STEP 6: Stop the program.

PROGRAM

```
#include<iostream.h>
int main()
{
double num1,num2,num3;
double sum,average;
cout<<"enter three numbers::";
cin>>num1>>num2>>num3;
sum=num1+num2+num;
average=sum/3;
cout<<"sum="<<sum<<endl;
cout<<"average="<<average<<endl;
return 0;
}
```

OUTPUT

```
Enter three numbers:
4
5
6
sum=15
average=5
```

RESULT

Thus the program has been executed successfully.

Ex. No: 4

CHARACTER PATTERN

Date:

AIM

To create a C++ program to Print Character Pattern using for loop.

ALGORITHM

STEP1: Start the program.

STEP 2: Declare and include the needed variables & functions.

STEP 3: Define the Execution Method to Print Character Pattern using for loop.

STEP 4: Save and compile the program

STEP 5: Run the program & Display the result.

STEP 6: Stop the program.

PROGRAM

```
#include <iostream>
 int main()
{
    int i, j;
    int rows = 5;
    char character = 'A';
    for (i = 0; i < rows; i++)
   {
    for (j = 0; j <= i; j++)
   {
    cout << character << " ";
     }
    cout << "\n";
    character++;
    }
    return 0;
}
```

OUTPUT

```
A
B B
C C C
D D D D
E E E E E
```

RESULT

Thus the program has been executed successfully.

Ex. No: 5

LCM CALCULATION

Date:

AIM

To create a C++ program to calculate the LCM of given n numbers.

ALGORITHM

STEP1: Start the program.

STEP 2: Declare and include the needed variables & functions.

STEP 3: Define the Execution Method to calculate the LCM of given n numbers.

STEP 4: Save and compile the program

STEP 5: Run the program & Display the result.

STEP 6: Stop the program.

PROGRAM

```
#include<iostream.h>
int gcd(int a,int b){
if(b==0)
return a;
return gcd(b,a%b);
}
int main(){
int a=7,b=5;
cout<<"LCM of"<<a<<"and"<<b<<"is"<<(a*b)\gcd(a,b);
return 0;
}
```

OUTPUT

LCM of 7 and 5 is 35

RESULT

Thus the program has been executed successfully.

Ex. No: 6

CALCULATING SQUARE & CUBE USING INLINE FUNCTION

Date:

AIM

To create a C++ program to calculate the square and cube value of given number using inline function.

ALGORITHM

STEP1: Start the program.

STEP 2: Declare and include the needed variables & functions.

STEP 3: Define the Execution Method to calculate the square and cube value of given number using inline function.

STEP 4: Save and compile the program

STEP 5: Run the program & Display the result.

STEP 6: Stop the program.

PROGRAM

```
#include<iostream.h>
#include<conio.h>
class power
{
public:
inline int square(int n)
{
return n*n;
}
inline int cube(int n)
{
return n*n*n;
}
};
void main()
{
int n,r;
power p;
clrscr();
cout<<"\n enter the number:\n";
cin>>n;
r=p.square(n);
cout<<"\n square of "<<n<<"="<<r<<endl;
```

```
r=p.cube(n);
cout<<"\n cube of "<<n<<"="<<r<<endl;
getch();
}
```

OUTPUT:

```
Enter the number:
3
Square of 3=9
Cube of 3=27
```

RESULT

Thus the program has been executed successfully.

Ex. No: 7

CALCULATING AREA & PERIMETER OF A RECTANGLE

Date:

AIM

To create a C++ program to calculate the area & perimeter of a rectangle.

ALGORITHM

STEP1: Start the program.

STEP 2: Declare and include the needed variables & functions.

STEP 3: Define the Execution Method to calculate the area & perimeter of a rectangle.

STEP 4: Save and compile the program

STEP 5: Run the program & Display the result.

STEP 6: Stop the program.

PROGRAM

```
#include<iostream.h>
#include<math.h>
class rectangle
{
int length;
int breadth;
public:
int getarea(int l,int b)
{
int area;
area=l*b;
cout<<"area="<<area<<"\n";
}
int getperimeter(int l,int b)
{
int peri;
peri=2*(l+b);
cout<<"perimeter="<<peri<<"\n";
}
};
int main()
{
int l,b;
rectangle r;
```

```
cout<<"enter length of rectangle";
cin>>l;
cout<<"enter breadth of rectangle";
cin>>b;
r.getarea(l,b);
r.getperimeter(l,b);
return 0;
}
```

OUTPUT:

```
Enter length of rectangle
3
Enter breadth of rectangle
5
Area=15
Perimeter=16
```

RESULT

Thus the program has been executed successfully.

Ex. No: 8

FUNCTION OVERLOADING

Date:

AIM

To create a C++ program to implement the function overloading.

ALGORITHM

STEP1: Start the program.

STEP 2: Declare and include the needed variables & functions.

STEP 3: Define the Execution Method to implement the function overloading.

STEP 4: Save and compile the program

STEP 5: Run the program & Display the result.

STEP 6: Stop the program.

PROGRAM

```
#include<iostream.h>
#include<conio.h>
class addition
{
public:
int sum(int a,int b)
{
return a+b;
}
int sum(int a,int b,int c)
{
return a+b+c;
}
};
int main(void)
{
addition obj;
cout<<obj.sum(20,15)<<endl;
cout<<obj.sum(81,100,10);
return 0;
}
```

OUTPUT:

35

191

RESULT

Thus the program has been executed successfully.

Ex. No: 9

OVERLOAD UNARY MINUS OPERATOR

Date:

AIM

To create a C++ program to overload the unary minus operator.

ALGORITHM

STEP1: Start the program.

STEP 2: Declare and include the needed variables & functions.

STEP 3: Define the Execution Method to overload the unary minus operator.

STEP 4: Save and compile the program

STEP 5: Run the program & Display the result.

STEP 6: Stop the program.

PROGRAM

```
#include<iostream.h>
#include<conio.h>
class minus
{
private:
int a,b,c;
public:
minus(){}//default constructor
minus(int A,int S,int C)
{
a=A;
b=S;
c=C;
}
void display(void);
void operator -();
};
inline void minus::display(void)
{
cout<<"\t a="<<a<<endl;
cout<<"\t b="<<b<<endl;
cout<<"\t c="<<c<<endl;
}
```

```
inline void minus::operator -()
{
a=-a;
b=-b;
c=-c;
}
void main(void)
{
clrscr();
minus M(5,10,-15);
cout<<"\n before activating the operator -()\n";
M.display();
-M;
cout<<"\n after activating the operator -()\n";
M.diplay();
getch();
}
```

OUTPUT:

Before activating the operator -()

a=5

b=10

c=-15

After activating the operator -()

a=-5

b=-10

c=15

RESULT

Thus the program has been executed successfully.

Ex. No: 10

CONSTRUCTOR

Date:

AIM

To create a C++ program to implement the constructor.

ALGORITHM

STEP1: Start the program.

STEP 2: Declare and include the needed variables & functions.

STEP 3: Define the Execution Method to implement the constructor.

STEP 4: Save and compile the program

STEP 5: Run the program & Display the result.

STEP 6: Stop the program.

PROGRAM

```
#include<iostream.h>
class Wall
{
private:
double length;
double height;
public:
Wall(double len,double hgt)
{
length=len;
height=hgt;
}
double calculateArea()
{
return length*height;
}
};
int main()
{
Wall wall1(10.5,8.6);
Wall wall2(8.5,6.3);
cout<<"area of wall1:"<<wall1. calculateArea()<<endl;
cout<<"area of wall2:"<<wall2. calculateArea();
return 0;
}
```

OUTPUT:

Area of Wall1 : 90.3

Area of Wall2 : 53.55

RESULT

Thus the program has been executed successfully.

Ex. No: 11

SINGLE INHERITANCE

Date:

AIM

To create a C++ program to implement the single inheritance

ALGORITHM

STEP1: Start the program.

STEP 2: Declare and include the needed variables & functions.

STEP 3: Define the Execution Method to to implement the single inheritance

STEP 4: Save and compile the program

STEP 5: Run the program & Display the result.

STEP 6: Stop the program.

PROGRAM

```
#include<iostream.h>
class base
{
public:
int x;
void getdata()
{
cout<<"enter the value of x=";
cin>>x:
}
};
class derive:public base{
private:
int y;
public:
void readdata()
{
cout<<"enter the value of y=";
cin>>y;
}
void product()
{
cout<<"product="<<x*y;
}
};
```

```
int main()
{
derive a;
a.getdata();
a.readdata();
a.product();
return 0;
}
```

OUTPUT:

```
Enter the value of X : 23
Enter the value of Y : 20
Product : 460
```

RESULT

Thus the program has been executed successfully.

Ex. No: 12

MULTILEVEL INHERITANCE

Date:

AIM

To create a C++ program to implement the multilevel inheritance.

ALGORITHM

STEP1: Start the program.

STEP 2: Declare and include the needed variables & functions.

STEP 3: Define the Execution Method to implement the multilevel inheritance.

STEP 4: Save and compile the program

STEP 5: Run the program & Display the result.

STEP 6: Stop the program.

PROGRAM

```
#include<iostream.h>
class animal{
public:
void eat(){
cout<<"eating..."<<endl;
}
};
class dog:public animal
{
public:
void bark(){
cout<<"barking..."<<endl;
}
};
class babydog:public dog
{
pubic:
void weep(){
cout<<"weeping...";
}
};
int main(void){
babydog d1;
```

```
d1.eat();
d1.bark();
d1.weep();
return 0;
}
```

OUTPUT:

```
eating.....
barking.....
weeping.....
```

RESULT

Thus the program has been executed successfully.

Ex. No: 13

MULTIPLE INHERITANCE

Date:

AIM

To create a C++ program to implement the multiple inheritance.

ALGORITHM

STEP1: Start the program.

STEP 2: Declare and include the needed variables & functions.

STEP 3: Define the Execution Method to implement the multiple inheritance.

STEP 4: Save and compile the program

STEP 5: Run the program & Display the result.

STEP 6: Stop the program.

PROGRAM

```
#include<iostream.h>
class A
{
public:
int x;
void getx()
{
cout<<"enter value of x:";cin>>x;
}
};
class B
{
public:
int y;
void gety()
{
cout<<"enter value of y:";cin>>y;
}
};
class C:public A,public B
{
public:
void sum()
{
cout<<"sum="<<x+y;
```

```
}
};
int main()
}
C obj1;
obj1.getx();
obj1.gety();
obj1.sum();
return 0;
}
```

OUTPUT:

```
Enter a value of X = 22
Enter a value of Y = 24
Sum = 46
```

RESULT

Thus the program has been executed successfully.

Ex. No: 14

STRING MANIPULATION

Date:

AIM

To create a C++ program to implement the string manipulation.

ALGORITHM

STEP1: Start the program.

STEP 2: Declare and include the needed variables & functions.

STEP 3: Define the Execution Method to implement the string manipulation.

STEP 4: Save and compile the program

STEP 5: Run the program & Display the result.

STEP 6: Stop the program.

PROGRAM

```
#include<iostream.h>
#include<conio.h>
  int main ()
 {
  string string1 = "Beginner ";
  string string2 = "to C++ ";
  string string3 = "Tutorials";
  string string4 = string1 + string2 + string3;
  int  len = string4.length();
  cout << string4 << endl;
  cout << "Length of string1 is: " << len <<endl;
  cout <<"Expert is at position " << string2.find("C++") <<endl;
  cout << "Part of string 2: " << string2.substr(3,8)<<endl;
  cout << "Replacing 'C++':  " << string4.replace(12, 17, "programming")<<endl;
  cout << "Insertion:  "<< string4.insert(0, " by Bon")<<endl;
  cout << "Erasing:  " << string3.erase(0,3)<<endl;
  getch();
  return 0;
 }
```

OUTPUT:

```
Beginner to Expert Tutorials
Length of string1 is: 28
C++ is at position3
Part of string 2: C++
Replacing 'C++':  Beginner to programming
Insertion:   by BonBeginner to programming
Erasing:  orials
```

RESULT

Thus the program has been executed successfully.

Ex. No: 15

MATH FUNCTIONS

Date:

AIM

To create a C++ program to implement the math functions.

ALGORITHM

STEP1: Start the program.

STEP 2: Declare and include the needed variables & functions.

STEP 3: Define the Execution Method to implement the math functions.

STEP 4: Save and compile the program

STEP 5: Run the program & Display the result.

STEP 6: Stop the program.

PROGRAM

```
#include <iostream.h>
#include <cmath.h>
int main ()
{
  int PI = 3.142;
  cout<< "cos(60) = " << cos ( 60.0 * PI / 180.0 )<<endl;
  cout<< "sin(60) = " << sin ( 60.0 * PI / 180.0 )<<endl;
  cout<< "tan(45) = " << tan ( 45.0 * PI / 180.0 )<<endl;
  cout<< "acos(0.5) = " << acos (0.5) * 180.0 / PI<<endl;
  cout<< "asin(0.5) = " << asin (0.5) * 180.0 / PI<<endl;
  cout<< "atan(1.0) = " << atan (1.0) * 180.0 / PI<<endl;
  cout<< "2^3 = " << pow(2,3)<<endl;
  cout<< "sqrt(49) = " << sqrt(49)<<endl;
  cout<< "ceil(3.8) = " << ceil(3.8)<<endl;
  cout<< "floor(2.3) = " << floor(2.3)<<endl;
  cout<< "fmod(5.3,2) = " << fmod(5.3,2)<<endl;
  cout<< "trunc(5.3,2) = " << trunc(2.3)<<endl;
  cout<< "round(4.6) = " << round(4.6)<<endl;
  cout<< "remainder(18.5,4.2) = " << remainder(18.5 ,4.2)<<endl;
  cout<< "fmax(100.0,1.0) = " << fmax(100.0,1.0)<<endl;
  cout<< "fmin(100.0,1.0) = " << fmin(100.0,1.0)<<endl;
  cout<< "fdim(2.0,1.0) = " << fdim(2.0,1.0)<<endl;
  cout<< "fabs(3.1416) = " << fabs(3.1416)<<endl;
  cout<< "abs(3.1416) = " << abs(3.1416)<<endl;
  cout<< "log(5) = " << log(5)<<endl;
  cout<< "exp(5.0) = " << exp(5.0)<<endl;
  cout<< "log10(5) = " << log10(5)<<endl;
  return 0;
}
```

OUTPUT:

cos(60) = 0.540302
sin(60) = 0.841471
tan(45) = 0.931596
acos(0.5) = 62.8319
asin(0.5) = 31.4159
atan(1.0) = 47.1239
2^3 = 8
sqrt(49) = 7
ceil(3.8) = 4
floor(2.3) = 2
fmod(5.3,2) = 1.3
trunc(5.3,2) = 2
round(4.6) = 5
remainder(18.5,4.2) = 1.7
fmax(100.0,1.0) = 100
fmin(100.0,1.0) = 1
fdim(2.0,1.0) = 1
fabs(3.1416) = 3.1416
abs(3.1416) = 3.1416
log(5) = 1.60944
exp(5.0) = 148.413
log10(5) = 0.69897

RESULT

Thus the program has been executed successfully.