Name	Mukund Kukreja	Year	2nd
Subject	OOPM	Class	AIML-1
Semester	III rd	Roll No.	0206AL211059

INDEX

Sr. No.	Experiment Description	Experiment Date	Submission Date	Remarks / Signature
1	Objects & classes	3/8/22		A
2	Arrays	10/8/22		A+
3	Single inheritance	17/8/22		
4	Multiple inheritance	24/8/22		B
5	Multi-level inheritance	7/9/22		B+
6	If_Else & If_Else_If	14/9/22		
7	Switch statement	28/9/22		A
8	Default constructor & Destructor	12/10/22		A
9	Parameterized constructor	19/10/22		A
10.	Function Overloading	26/10/22		A
11.	Function Overriding	2/11/22		
12.	Friend function	9/11/22		A+
13.	Virtual Function	16/11/22		A+
14.	Static function	23/11/22		A+
15.	Exception Handling	30/11/22		A+

Output 1 >> Data member value = 123
Member function called!

Date : 29/9/22
Page : 1

Experiment - 1

Write a program to illustrate the concept of objects & classes.

```
# include <iostream>
using namespace std;

class MyClass { // class definition
        public:
            int dataMember = 123;
            void memberFunction ( ) {
                cout << "Member function called! " << endl;
                }
};  // terminating class definition

int main ( ) {
        MyClass myObj;  // initializing an object of MyClass

    cout << " Data member value = " << myObj. dataMember << endl;
    // accessing public datamember

    myObj. memberFunction ( );  // calling public member function
    return 0;
}
```

Output 2 >> Enter the size of your array: 3
Enter Element 1: 2
Enter Element 2: 4
Enter Element 3: 6
Statically defined array:
10
20
30
40
50
User array:
2
4
6

Experiment - 2

Write a program to illustrate the concept of arrays.

```
#include <iostream>
using namespace std;

int main(){
    int arr[5] = {10, 20, 30, 40, 50}; // defining an array statically.

    int size;
    cout << "Enter the size of your array: ";
    cin >> size;

    int userArr[size]; // declaring an array dynamically

for (int i=0; i<size; i++){
    cout << "Enter Element " << i+1 << ":";
    cin >> userArr[i];
} // taking elements for array as user input

        cout << "Statically defined array: " << endl;
    for (int n=0; n<5; n++){
        cout << arr[n] << endl;
    } // displaying arr[]

cout << "User array: " << endl;
for (int k=0; k<size; k++){
        cout << userArr[k] << endl;
    } // displaying userArr[]

return 0;
}
```

Output 3 >> Base Class variable = 345
Base class function called!
Derived class variable = 682
Derived class function called!

Experiment - 3

Write a program to illustrate the concept of single inheritance

```
#include <iostream>
using namespace std;

class BaseClass{
    public:
        int baseVar = 345;
        void baseFunc(){
            cout << "Base class function called! " << endl;
        }
};

class DerivedClass : public BaseClass{
    public:
        int derivedVar = 682;
        void derivedFunc(){
            cout << "Derived class function called! " << endl;
        }
};

int main(){
    DerivedClass d; // creating object of derived class

cout << "Base Class variable = " << d.baseVar << endl;
d.baseFunc(); // accessing base class members from derived class object
cout << "Derived class variable = " << d.derivedVar << endl;
d.derivedFunc();
return 0;
}
```

Output 4 >>

```
I am an animal!
I can swim!
I can walk!
I am a crocodile!
```

Experiment - 4

Write a program to illustrate the concept of multiple inheritance

```
# include <iostream>
using namespace std;

class Swim {
    public : void swim(){
                cout << " I can swim! " << endl;
             }
};

class Walk {
    public : void walk(){
                cout << " I can walk! " << endl;
             }
};

class Animal {
    public : Animal() { cout << " I am an animal! " << endl; }
};

class Crocodile : public Animal, public Swim, public Walk {
    public : Crocodile(){ swim(); walk();
                cout << " I am a crocodile! " << endl; }
};
int main(){
    Crocodile croc;
    return 0;
}
```

Output 5 >>

Class A Constructor invoked!
Class B constructor invoked!
Class C constructor invoked!
a = 123
b = 234
c = 456

Experiment-5

Write a program to illustrate the concept of multi-level inheritance

```
#include <iostream>
using namespace std;

class A{
    public:
        int a=123;
        A(){ cout<<" Class A constructor invoked! "<<endl; }
};

class B : public class A{
        public:
            int b=234;
            B(){ cout<<"Class B constructor invoked! "<<endl; }
};

class C : public class B{
        public:
            int c=456;
            C(){ cout<<Class C constructor invoked! "<<endl;}
};

int main(){
        C objOfC;
    cout<<" a = "<<objOfC.a<<endl;  // accessing members
    cout<<" b = "<<objOfC.b<<endl  // of class A & B from
    cout<<" c = "<<objOfC.c<<endl; // object of class C
    return 0;
}
```

Output 6 » Enter your age to check if you are eligible for voting : -12
Enter a valid age!

» Enter your age to check if you are eligible for voting : 16
You are NOT eligible for voting!

» Enter your age to check if you are eligible for voting: 19
You are eligible for voting!

Experiment-6

Write a program to illustrate the concept of IF_ELSE and IF_ELSE_IF control statements.

```
#include <iostream>
using namespace std;

int main() {
    cout << "Enter your age to check if you are eligible for
            voting : ";
    int age;
    cin >> age;

    if (age < 1) cout << "Enter a valid age!" << endl;
    else if (age < 18) cout << "You are NOT eligible for voting!" << endl;
    else cout << "You are eligible for voting!" << endl;

    return 0;
}
```

Output 7 >> The menu is:
1. Pizza
2. Burger
What do you want to order?: 17
Enter a valid number from menu!

>> The menu is:
1. Pizza
2. Burger
What do you want to order?: 1
Sorry! Pizza is currently unavailable...

>> The menu is:
1. Pizza
2. Burger
What do you want to order?: 2
One Burger coming right up!

Experiment - 7

Write a program to illustrate the concept of switch statement

```
#include <iostream>
using namespace std;

int main() {
    cout << "The menu is: " << endl;
    cout << " 1. Pizza " << endl;
    cout << " 2. Burger " << endl;
    cout << " What do you want to order? : ";
    int order;
    cin >> order;

    switch (order) {
      case 1:
        cout << " Sorry! Pizza is currently unavailable... " << endl;
        break;

      case 2:
        cout << " One Burger coming right up! ";
        break;

      default:
        cout << " Enter a valid number from menu! ";
        break;
    }
    return 0;
}
```

Output 8 >> Default Constructor called!
Doing some work with the object!
Destroying object after work is done!

Experiment 8

Aim:- Write a program to illustrate the concept of and Destructor

```
#include <iostream>
using namespace std;

class MyClass {
    public:
        MyClass () { // default constructor
            cout << " Default constructor called! " << endl;
        }
        void doSomething (){
            cout << " Doing some work with the object! " << endl;
        }

        ~MyClass () { // destructor
            cout << "Destroying object after work is done! " << endl;
        }
};

int main () {
    MyClass obj;
    obj.doSomething ();
    return 0;
}
```

Output 9 >> Parameterized Constructor called!

Hi! Harish

Experiment - 9

Write a program to illustrate the concept of parameterized constructor

```
#include <iostream>
using namespace std;

class Person {
    public: string n;
public: Person( string name ){ // parameterized constructor
        cout << "Parameterized Constructor called! "<< endl;
        n = name;
    }
public: void greet ( ){
        cout << "Hi! " << n << endl;
    }
};

int main ( ){
    Person p1("Harish");
    p1.greet( );
    return 0;
}
```

Output 10 >>

```
Area of circle: 12.56   of radius: 2
Area of square: 4       of side: 2
Area of triangle: 6     of base: 3 and height: 4
Area of retangle: 12    of length: 3 and breadth: 4
```

Experiment - 10

Write a program to illustrate the concept of function overloading.

```
#include <iostream>
using namespace std;

class Shape {
    public:
        void Area (float dim){
        cout << "Area of circle: "<<dim * 3.14 * dim << " .of radius: "<< dim <<endl;
        cout << "Area of square: "<< dim * dim << "of side : " << dim << endl;
    }

    void Area (float dim1 , float dim2){
    cout << "Area of triangle: " << dim1 * dim2 * 0.5 << " of base: "
        << dim1 << "and height: " << dim2 << endl;

    cout << "Area of rectangle: " << dim1 * dim2 << " of length: " << dim1
        << " and breadth: " dim2 << endl;
    }
};

int main(){
    Shape(). Area (2);
    Shape(). Area (3,4);
    return 0;
}
```

Output >> Meow!
Woof!
This animal produces sound

Experiment-11

Write a program to illustrate the concept of function overriding

```
#include <iostream>
using namespace std;

class Animal { // base class
    public: void speak() { cout << "This animal produces sound \n"; }
};

class Cat : public Animal { // derived class
  public: void speak() { cout << "Meow!" << endl; }
};

class Dog : public Animal { // derived class
    public: void speak() { cout << "Woof!" << endl; }
};

class Horse : public Animal { }; // derived class

int main() {
    Cat c;
    Dog d;
    Horse h;

    c.speak();
    d.speak();
    h.speak();
    return 0;
}
```

Output 12 » Mean Value = 32.5

Date :
Page : 12

Experiment - 12

Write a program to illustrate the concept of Friend function.

```
#include <iostream>
using namespace std;

class sample{
    int a,b;
    public:
    void setValue() { a=25 ; b=40; }
    Friend float mean(sample s);
};

float mean (sample s){
    return float (s.a + s.b) / 2.0;
}

int main ( ) {
    sample X ;
    X. setValue ();
    cout << " Mean Value = " << mean(X) << endl;
    return 0;
}
```

Output 13 >> print derived class
show base class

Experiment -13

Write a program to illustrate the concept of virtual function.

```
#include <iostream>
using namespace std;

class Base {
    public:
    virtual void print() { cout << "print base class \n"; }
    void show() { cout << "show base class \n"; }
};

class Derived : public Base {
    public:
    void print() { cout << "print derived class \n"; }
    void show() { cout << "show derived class \n"; }
};

int main() { Base *bptr;
             Derived d;
             bptr = &d;
             bptr -> print();
             bptr -> show();
             return 0; }
```

Output 14 » This method was invoked without making an object

Experiment - 14

Write a program to illustrate the concept of static function.

```
#include <iostream>
using namespace std;

class MyClass {
    public:
        static void print() { // static member function
            cout << "This method was invoked without making
                an object" << endl;
        }
};

int main() {
    MyClass::print(); //invoking a static member function
                      //by using scope resolution operator
    return 0;
}
```

V. good

Date :
Page. 15

Experiment – 15

Write a program to illustrate the concept of exception handling.

```
#include <iostream>
using namespace std;

int main(){

try{    // code that can cause an exception is written inside try block
    int input;
cout << " Enter a number between 0 and 10: ";
cin >> input;

if(0 <= input && input <= 10) cout << "Thank You for your input!\n";
else throw (input);  // throws an exception
}
catch(int num){ // the exception is catched here.

    cout << " Only 0 to 10 numbers are valid input! \n";
    cout << " You entered: " << num;
}

return 0;
}
```

Output 15 [exception is caused] >> Enter a number between 0 and 10 : 23
Only 0 to 10 numbers are valid input!
You entered : 23

[exception is Not caused] >> Enter a number between 0 and 10 :
Thank You for your input!