Ref No: EX/IT/PC/B/T/214/2023 (S)

B.E in INFORMATION TECHNOLOGY

2ND YEAR, 1ST SEMESTER SUPPLEMENTARY EXAMINATION, 2023

OBJECT ORIENTED PROGRAMMING

Time: 3 hours **Full Marks:** 100

CO1 **[15 MARKS]**	1. a) Which of the following function prototypes are valid? State reasons. For each of the valid prototypes, show how many ways the function can be invoked. *int sum(int i=1, int j, int k);* *int sum(int i=1, int j=2, int k);* *int sum(int i=1, int j, int k=3);* *int sum(int i, int j=2, int k=3);* *int sum(int i, int j=2, int k);* *int sum(int i, int j, int k=3);* b) State 2 circumstances where ***this*** operator is used. Illustrate with suitable code snippets. c) Distinguish between normal function and inline function. State some scenarios where a function is not executed as ***inline*** even if it is declared as inline. d) Can a function call statement appear on the left hand side of an assignment operator? If no, state why. If yes, show how. [6+3+(2+1)+3=15]
CO2 **[30 MARKS]**	2. a) Consider the following code snippet having the definition of a class ***Number***. Keeping the class definition unchanged, define all the methods and constructor outside the class definition and show how they are invoked from ***main()*** method. Discuss the outputs. Point out the drawback of the code and state how this drawback can be overcome. *class Number* *{ int a;* *public:* *Number(int=0);* *void add(Number);* *friend void swap(Number, Number);* *void show();* *};* b) Justify the truth or falsity of each of the following statements. Provide code snippets, where/if necessary, for justification. i) A class can have several constructors and destructors. ii) A copy constructor of a class takes the photocopy of object of that class. iii) A non-static member method of a class can access only the non-static member variables of that class. iv) Method overloading is also known as early binding. v) A constant member function of a class can access only the constant member variables of a class. c) Which property of OOP is not maintained by friend functions and why? Write a complete C++ program to define a class ***Time*** with 2 data members: ***hour*** and ***minute*** and create two objects from that class T1 and T2. Now define a friend function of ***Time*** class to perform addition between the objects T1 and T2 and print the result as follows. ***Example:*** 3 hr 40 min (***T1***) + 2 hr 30 min (***T2***)= 6 hr 10 min (***result***) [(5+1+2)+(3x5)+(2+5)=30]

CO3

[20 MARKS]

3. a) Suppose a class ***Employee*** has a private member variable ***basic_sal*** and a member function ***get_sal().*** Two subclasses ***Manager*** and ***Clerk*** have been derived from it. These two types of employees receive 40% and 30% allowances respectively on their basic monthly salary. Show with a suitable code how to compute the total salary of these two types of employees. What type of inheritance does it refer?

b) Assume a class ***Base*** with a parameterized constructor and a destructor. Create a derived class ***Child*** inheriting the ***Base*** class having its own constructor and destructor. Show (with code snippet) how the constructor of the ***Base*** class is invoked through ***Child*** class. State in what order these constructors and destructors of the two classes are invoked.
Discuss in this connection, the drawback of such destructor and how this drawback can be overcome.

c) Suppose two classes ***Base1*** and ***Base2*** have a public member function ***display()*** each within them. A ***Child*** class has been inherited from these two classes. Show (with proper code) how the ***Base2*** class version of the ***display()*** function is invoked through the ***Child*** class object. What type of inheritance does it refer?

d) Distinguish between virtual function and pure virtual function with code snippets.

[(5+1)+(4+2+2)+(3+1)+2=20]

CO4

[20 MARKS]

4. a) Consider the following class ***Sample.*** Modify the class with suitable code to carry out the operations as mentioned in the *main()* method. Indicate the value of the variable ***t*** which will be printed.

```
class Sample
{int a;

public:
//add suitable member methods
};
```

```
int main()
{Sample s1(5), s2, s3(10);
s2=s1+(++s3);
s2=s3+3;
int t=s2;
cout<<"t="<<t;
//invoke display() method to print the values of
the member variables of s1, s2 and s3
}
```

b) Consider the following class ***Sample.*** Update its definition to perform the tasks as specified in ***main().*** Clearly indicate which portion of the class is dedicated for which task. Finally discuss the output.

```
class Sample
{   int s;
};
```

```
int main()
{ Sample ob1(3), ob2(4);
 ob1+=ob2;
ob1++.show();
ob2=10+ob1;
cout<<ob1<<ob2;
}
```

c) Consider a namespace NS having the declaration of a function ***display()*** and the definition of a class ***Sample.*** This ***Sample*** class is also having the declaration of a member function with the same signature as that of the ***display()*** method. Now show how these two versions of the ***display()*** method can be defined outside the namespace definition.

[8+8+4=20]

CO5

[15 MARKS]

5. a) Illustrate how exceptions of integer, float, character and string types are thrown from one *try* block and caught in different *catch* blocks. Show how to replace these multiple *catch* blocks by a single one.
b) When does the need of template specialization arise? Show with suitable code snippet.
c) Show (with a snippet) how an exception can be re-thrown?
d) How can you supply default parameter/s in template functions? Show with suitable code snippet.

[(4+2)+3+3+3=15]