Ref No: EX/IT/PC/B/T/214/2024

B.E in INFORMATION TECHNOLOGY

2ND YEAR, 1ST SEMESTER EXAMINATION, 2024

OBJECT ORIENTED PROGRAMMING

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

Answer all parts of a question together in one place. Do not scatter the answers.

CO1 **[20 MARKS]**	1. a) Fill in the blanks with appropriate phrases. Hence justify the validity of each of them. Provide suitable arguments and proper syntax/code snippets where necessary. i) Default values in function parameters are always supplied from _____ to _____. ii) A _____ cannot appear on the left hand side of _____operator unless _____. iii) If variables are passed to a function by _____, modifications to them are not reflected in _____. To retain the modifications, we have to _____. iv) Both malloc() and calloc() require _____. However, it is not required in_____. v) Recursive functions are executed in _____ of the memory. Every recursive function must have a _____. Otherwise it will lead to _____. vi) The _____keyword is just a request to the _____. However, this request may be ignored in some cases like_____. b) Write a recursive function that finds out the highest digit from a number N. N will be given by the user as input to the function. Next check (using loop or recursion) whether this highest digit belongs to the Fibonacci series or not. Example: Input N-> 5873. Highest digit=8. 8 belongs to the Fibonacci series. Or, Create an array of 10 integers dynamically. Then pass the array to a recursive function that finds and returns the highest element of the array. Next find the highest digit from this element (using recursion). [(2x6)+8=20]
CO2 **[20 MARKS]**	2. a) Justify the truth/falsity of each of the following statements. Provide suitable arguments in support of your answers along with suitable code snippets where/if required. i) The static member variables of a class cannot be accessed within the non-static member functions. ii) Two functions having prototypes ***void sum(int, int)*** and ***int sum(int, int)*** cannot be overloaded. iii) For cascading call of member methods of a class, the methods can return anything other than **void.** iv) A constant member function of a class cannot be invoked by a non-constant object of that class. v) A friend function, if declared within a class, must be defined outside the class using **::** operator. b) Fill in the blanks with appropriate phrases. Hence justify the validity of each of them. Provide suitable arguments and code snippets where necessary. i) A constructor of a class can accept any type of data as parameters except _____. ii) The _____variables of a class are object-independent because _____. iii) The _____and _____ functions of a class cannot access ***this*** pointer. iv) A class is a/an _____ entity, whereas an object is a/an_____ entity. v) In order to assign different values to an array of objects using constructor, we have to_____. [(2x5)+(2x5)=20]

[Turn over

CO3 **[20 MARKS]**	3. a) Discuss the drawback of each of the following. Hence discuss how the drawbacks can be overcome. Use suitable code snippets in support. i) Multiple inheritance ii) Normal destructor iii) Hybrid inheritance b) Distinguish among each of the following triples. Use suitable code snippet in support of your answer. i) Private, protected, and public mode of inheritance ii) Abstract class, derived class, and friend class c) Suppose a class ***Employee*** has a private member variable ***basic_sal***, a suitable constructor to set its value and a only one member function ***get_basic_sal()*** to retrieve it. Two subclasses ***Manager*** and ***Clerk*** have been derived from it having their own parameterized constructors to set the value of ***basic_sal***. These two types of employees receive 40% and 30% allowances respectively on their basic monthly salary. These allowances should be calculated by defining a member function ***cal_allowance()*** within each of the derived classes. Show with a suitable code how to compute the total salary of these two types of employees **strictly** using pointers of ***Employee*** class only. Assuming there are 5 managers and 20 clerks, create an array of pointers of ***Employee*** class to compute the total expenditure of the company per month for disbursing the salary of all the employees. **(No member functions of derived classes should be called using derived class objects).** What type of inheritance does it indicate? [(2+2+2)+(3+3)+(4+3+1)=20]
CO4 **[20 MARKS]**	4. a) Consider the following two classes ***A*** and ***B***. Complete their definitions to execute the statements specified in ***main()***. Clearly indicate which portion of the class is dedicated for which task. Finally discuss the output. **The order of the class definitions should not be changed and use of friend functions/classes should be avoided.** *class A* *{ int a;* *};* *class B* *{ int b;* *};* *int main()* *{ A a1,a2;* *B b1(5),b2,b3;* *(b1++)->show();* *a1=b1;* *cout<<a1<<endl;* *b3=b2(5);* *cout<<b2<<" "<<b3<<endl;* *a2=(b2+b3)-5;* *cout<<a2<<endl;* *if(a1!=a2)* *cout<<"Not equal";* *else* *cout<<"Equal";* *}*

b) Consider the following ***Sample*** class. Complete it with appropriate code to perform the operations as directed in the ***main()*** function. Clearly indicate which portion of the class is dedicated for which task. Finally discuss the output with proper logic.

```
class Sample
{ int s;

};
```

```
int main()
{  Sample ob1(4), ob2(5), ob3;
 ob3=ob1/=2;
cout<<ob1<<"  " <<ob3<<endl;
ob3=ob1*(ob2+5);
cout<<ob1<<"  "<<ob3<<endl;
ob2=50 – (++ob3)*ob1;
cout<<ob1<<"  "<<ob2<<"  "<<ob3<<endl;
}
```

c) Consider the following namespaces. State the output of each and every invocation of the ***show()*** method with brief explanation.

```
namespace ns1
{
 int a,b;
 void set (int x, int y)
 { a=x+y; b=y-x;
 }
void show()
{
cout<<a<<"  "<<b<<endl;
}
namespace ns2
{
 int a=5, b=2;
 void set (int x, int y)
 { a=y-x; b=x+y;
 }
void show()
{
cout<<a<<"  "<<b<<endl;
}
} /*end of ns2*/
}/*end of ns1*/
```

```
int main()
{
using namespace ns1;
a=10;
b=20;
ns2::show();
ns2::set(7,10);
show();
set(8,4);
ns2::show();
ns2::set(6,8);
show();
set(2,5);
ns2::show();
}
```

Or,

Discuss the functionality of the operator ***dynamic_cast*** with respect to RTTI. Provide suitable example of code snippet to explain.

[8+7+5=20]

CO5

[20 MARKS]

5. a) Justify the truth/falsity of each of the following statements. Provide suitable arguments in support of your answers along with suitable code snippets where required.

i) If an exception is not properly handled within a catch block, then the program is sure to be abruptly terminated before successful completion.

ii) After executing the statements in the catch block, the program control again returns to the try block from where the exception was thrown.

iii) Specialization of function/class template is not required for primitive data types.

b) Fill in the blanks with appropriate phrases. Hence justify the validity of each of them with proper logic and code snippets.

i) A _____ block can never appear without a _____ block.
ii) A function may or may not throw an exception. However, for preventive measure, we must have to ensure that _____.
iii) If we are not sure what type of data is being thrown within a try block, we may use _____. However, it must be _____ in case _____.

c) Consider a function template having the following signature:

```
template <class T=int, int N=5>
void print(T a)
{
 for(int i=1;i<=N;++i)
 cout<<a<<"  ";
 cout<<endl;
 }
```

Now identify which of the following function call statements are valid and which are invalid. Provide supporting reasons for each of them. Discuss the output for the valid ones.

```
print<4>('A');
print<double> (92.56);
print<int>(93,4);
print<>(94.56F);
print<int,4> (95);
print<double,5> (96.05);
print<char,4>(97);
print<>(98);
```

Or,

Write a complete C++ program to open a file ***a.txt*** in input mode. Now select all the words in this file which contain atmost 1 vowel and write them into another file ***b.txt***. Count how many such words are present over there.

[(2x3)+(2x3)+8=20]

Course outcomes:

CO1: Recognise and **illustrate** the procedural enhancements of object-oriented programming languages over procedural languages.

CO2: Explain, illustrate and **recognise** the basic features of classes and objects.

CO3: Illustrate the extended features of OOP (Inheritance, Polymorphism) and **apply** them in practical problem solving.

CO4: Explain and **illustrate** RTTI, Namespace and Operator overloading.

CO5: Demonstrate I/O, exception handling and generic programming.