Ref No: EX/IT/T/211/2019

B.E in INFORMATION TECHNOLOGY

2ND YEAR, 1ST SEMESTER 2019

OBJECT ORIENTED PROGRAMMING

Time: 3 hours *Question No. 1 and 2 are compulsory* **Full Marks:** 100

CO1:

1. a) Distinguish between ***malloc()*** function and ***new*** operator. Specify the syntax of creating an array of 10 integers separately using these two.
b) Explain the output of the following code snippet with proper reason. Show another 2 scenarios where scope resolution operator is used.

```
int k=1;
int main()
{int k=2;
 {int k=3;
 cout<<"k="<<k;
 cout<<"::k="<<::k;
 }
cout<<"k="<<k;
cout<<"::k="<<::k;
}
```

c) Distinguish between call by address and call by reference with suitable example of code snippets.
d) Can a function call statement appear on the left hand side of an assignment operator? Show with suitable code snippet. Which of the following prototypes is/are valid for functions with default parameters? State reason.

```
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);
```

[(3+3)+(3+3)+3+(3+2)=20]

CO2:

2. a) "*A constant member function can only read the member variables of the class in which it is defined.*"- justify the truth or falsity of the statement with supporting arguments.
b) Consider the following class ***Sample***. Define suitable constructors for the class in support of the statements in the main() method. Also indicate which statements in the main() method do not invoke any constructors.

```
class Sample
{int a;
public:
//define suitable constructors
};
```

```
int main()
{Sample s1(10), s2;
Sample s3=s1;
s2=s3;
Sample s4(s2);
Sample s5=Sample(20);
}
```

c) Explain the output of the following code snippet with proper reasons. Add appropriate code to set the initial value of the static variable *j* to 1.

```
class Sample
{ static int j;
public:
Sample()
{j++;
}
void display()
{cout<<"j="<<j;
}
};
```

```
int main()
{Sample s1;
s1.display();
Sample s2;
s2.display();
Sample s3,s4,s5;
s4.display();
s5.display();
Sample s[10];
for(int i=0;i<10;i++)
s[i].display();
}
```

d) Which property of OOP is not maintained by friend functions and why?

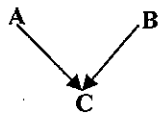
Write a complete C++ program to define a class ***Time*** with 3 data members ***hour, minute*** and ***second*** and create 2 objects from that class T1 and T2. Now define a friend function of ***Time*** class to perform addition between the objects T1 and T2 and print the result as follows.

Example: 3 hr 40 min 30 sec (***T1***) + 2 hr 30 min 40 sec (***T2***)= 6 hr 11 min 10 sec (***result***)

[2+(5+1)+4+(2+6)=20]

CO3:

3.a) Which type of inheritance does the following diagram indicate? Suppose both the A and B classes have a function with the same name as ***fun()***. How can you call the ***fun()*** function of the class B using an object of class C? Write appropriate code snippet to show.



b) What do you mean by diamond problem in inheritance? How it can be eliminated? Show with proper code snippet. Explain function overriding with respect to multi-level inheritance.

c) What is the need of declaring virtual destructors? How can you declare a pure virtual destructor? Provide suitable code snippets to illustrate both of these two.

State the difference between pure virtual function and pure virtual destructor.

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

Or,

4. a) Where lies the difference between ***protected*** and ***private*** access specifiers?

Suppose a class B is derived from a class A separately in private and public modes. State how the data members and member functions declared in private, protected and public section of the class A are inherited in class B.

b) What do you mean by run-time polymorphism? Show with a suitable code snippet. How does it differ from compile time polymorphism? State 2 cases of compile time polymorphism.

c) What type of inheritance does the following diagram indicate? Suppose the class A has a parameterized constructor. How can you invoke that constructor by creating an object of class B?

Next assume that another class D has been inherited in public mode from the classes B and C. Then how that constructor of the class A will be invoked by an creating object of class D? Assume that both the classes B and C have a one-parameterized constructor each of their own.

Provide supporting code snippets for both the above two cases.

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

CO4:

5. 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) What do you mean by namespace extension? Discuss with suitable example. State the difference between ***using declaration*** and ***using directive*** with necessary snippet.

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

[10+(1+2+3)+(1+3)=20]

Or,

6. a) Consider the following 2 classes. Modify them with appropriate code to implement the functionalities as mentioned in the main method.

```
class Sample
{
int s;
public:
//add suitable member methods
};
class Test
{
int t;
public:
//add suitable member methods
};
```

```
int main()
{
Sample s1, s2(10);
Test t1=15;
s1=t1;
s1->display();

Test t2(20,25), t3;
t3=t2(15,30,45);
cout<<t3;
}
```

b) Complete the following code fragment with necessary modifications in the class ***Array*** to perform the tasks as mentioned in the main method.

```
class Array
{
char *arr;
char ch;
public:
//add suitable member methods
};
```

```
int main()
{
Array a(15); //creates an array of 15
characters and initializes it
a[4]='c';

//invoke display() method to show the
contents of the array

char *ptr=new ('$') char; //initializes the
member variable ch to $
}
```

c) What is meant by unnamed namespace? Consider the following 2 namespaces. State the output of each and every invocation of the ***disp()*** method for the 3 different places with brief explanation.

```
namespace ns1
{
int a,b;
 void disp()
 {cout<<a+b;
 }
namespace ns2
{
int a,b;
 void disp()
 {cout<<a+b;
 }
}
}
```

```
int main()
{
ns1::a=10;
ns1::b=20;
ns1::disp();

ns1::ns2::a=30;
ns1::ns2::b=40
ns1::disp();

using namespace ns1::ns2;
a=50; b=60;
disp();
}
```

[10+5+(2+3)=20]

CO5:

7. a) Write a C++ program to open a file in *output* mode and writes one line at a time. The process will continue till the user enters -1. State the functionalities (with syntax of each) of ***tellg()***, ***seekg()*** and ***read()*** functions.
b) Write overloaded template functions to implement the following functionalities.

```
int main()
{sum(2,3,4.5);
sum(2.5,4.5,3);
sum(1,2,3)
sum(1.5,2.5);
}
```

c) When does the need of template specialization arise? Show with suitable code snippet.
d) What is the functionality of generalized ***catch*** statement? Where is it placed?

[(6+3)+5+4+2=20]

Or,

8. a) Write a complete C++ program to open a file in *input* mode and reads one character at a time. The process will continue till the *end of file* character is reached. State the functionalities (with syntax of each) of ***tellp()***, ***seekp()*** and ***write()***functions.
b) How can you supply default parameter/s in template functions? Show with suitable code snippet.
c) Show with appropriate example of code how an object of a class can be thrown during exception handling.
d) Show (with a snippet) how an exception can be re-thrown?

[(6+3)+4+4+3=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.