Ref No: EX/IT/PC/B/T/214/2023

B.E in INFORMATION TECHNOLOGY

2ND YEAR, 1ST SEMESTER EXAMINATION, 2023

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) Distinguish among following triples: (Any 3)
i) Call by value, call by reference and call by address
ii) malloc(), calloc() and new
iii) Normal function, inline function and macro
iv) Direct recursion, indirect recursion and tree recursion

b) Write a complete program that allocates an array of N number of integers dynamically. Then pass the array pointer to a function that finds out the pair of array elements whose summation is greater than or equal to S. (N and S will be given by the user at runtime)

E.g, Suppose an array contains 1,2,3,4. The pair of elements whose sum is greater than or equal to 5 are (1,4), (2,3), (2,4), and (3,4). The other pairs (4,1), (3,2), (4,2) and (4,3) **should not** be considered again.

c) Illustrate the use of :: operator in regard to non-OOP feature.

[(3x3)+8+3=20]

CO2 [20 MARKS]

2. a) Assume a class ***Sample*** having the following definition. Indicate which statements will report error/s. Provide supporting reasons for each of them.
Hence correct the entire code and predict the output with proper reasons based on your written corrected code. (The member variables *s, c* and *x* **should not** be changed)

```
class Sample
{
 static int s;
 const int c;
 int x;
public:
Sample(int y=0)
{
 c=y++;
 s=x=y;
}
void increment ()
{
s++; x++;
}
static void show()
{
cout<<s<< "  "<<x<< "  "<<c<<endl;
}
};
```

```
int main()
{ Sample ob1(5), ob2(6);
  ob1.show();
 ob2.show();
  Sample::increment();
  ob1.show();

  Sample ob[3];
 for(int i=0;i<3;i++)
  {ob[i].increment();
  ob[i].show();
 }
}
```

[Turn over

	b) Assume a class ***Weight*** as the following definition. Now complete the class definition in order to properly execute the statements in the ***main()*** method. Discuss the output.
	`class Weight { int kilogram, gram; };`
	`int main() { Weight w1(40,900), w2(60,700), w4; Weight w3=w1; w4=add(w3,w2); w4.show(); w4=w2.subtract(w3); w4.show(); }`
	
	c) Fill up the blanks with appropriate phrases. Hence justify the validity of each of the complete statements. Provide code snippets where/if necessary in support of your arguments.
	
	i) The main difference between a structure and a class is ______________. It makes ________ more suitable than ___________ for ensuring ____________ property.
	ii) For cascading function call, all the functions except the ________ must return______________.
	iii) A constant member method is __________ type of function, but it can ___________ the _________ type of member variables.
	iv) If within a class X, another class Y is declared to be friend, then all the member functions of______ can access the private data members of_______. This violates the ___________property of OOP.
	
	[(3+3)+6+(2x4)=20]

CO3 [20 MARKS]	3. a) Fill up the blanks with appropriate phrases and hence justify each of the complete statements with proper reasons. Provide code snippets where necessary to validate your answers. i) Run time polymorphism is known as __________ binding and it can be achieved by defining a _____________ member function in the Base class. ii) A pure virtual destructor ____________ outside the class and it makes a class ____________. iii) Hybrid inheritance suffers from ____________and it can be overcome by _____________. iv) The ___________ member variables of a Base class can be accessible within the Derived class but not outside the classes. Thus they have less strict accessibility than __________ data members. v) A ___________ function of the Base class _______________ in the Derived class. Otherwise, the Derived class cannot be instantiated. vi) For function overriding, the signatures of the Base class and Derived class function must be _________. Otherwise instead of overriding it becomes ____________. vii) In case of single inheritance, the Base class version of an overridden method can be invoked by ___________ from Derived class. viii) The Base class is/has ________________ of any newly added member variables of the Derived class. This property ensures _____________. ix) A default constructor of the Base class is invoked ___________. However, its parameterized constructor must be invoked ____________ from the Derived class. x) Multiple inheritance faces a problem when _______________ and it can be overcome by________. [(2x10)=20]
CO4 [20 MARKS]	*4.* a) 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 with reason.

```
class Weight
{
 int kilogram, gram;
};
```

```
int main()
{
  Weight w1(40,900), w2(60,700), w4;
  Weight w3=w1;
  w4=add(w3,w2);
  w4.show();
  w4=w2.subtract(w3);
  w4.show();
}
```

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

```
int main()
{ Sample ob1(3), ob2(4), ob3;
 ob3=ob1+=ob2;
ob3++.show();
ob2=10 + ++ob1;
cout<<ob1<<ob2;
}
```

b) Consider the following two classes *A* and *B*. Complete their definitions as per the statements specified in *main()* function. The order of the classes should remain same. Clearly indicate which portion of the class is dedicated for which task. Finally discuss the output.

```
class A
{  int a;
};

class B
{ int b;
};
```

```
int main()
{ A oba1(5), oba2;
oba2*=oba1+10;
B obb1=oba2;
obb1->show();
B obb2=obb1(10);
obb2->show();

A *ptr=new A(5);
if(oba1==*ptr)
oba1.display();
else
ptr->display();
delete ptr;
}
```

c) How many types of operators are there in RTTI? Illustrate the use of *dynamic_cast* operator.

Or,

Consider the definitions of the namespaces NS1 and NS2. Discuss the output with reasons.

```
namespace NS1
{
 int a, b;
   void f(int x, int y)
  {
   a+=++x;
   b+=y++;
  cout<<a<< endl<<b;
  }
 namespace NS2
  {
  int x, y;
  void g()
   {
   a+=++x;
   b+=--y;
  cout<<a<<endl<<b;
 cout<<x<<endl<<y;
   }
  }
}
```

```
int main()
{
using namespace NS1;
a=10, b=20;
f(10,20);
using NS2::x;
using NS2::y;
x=5; y=6;
NS2::g();
}
```

[7+8+5=20]

CO5 [20 MARKS]

5. a) Fill up the blanks with appropriate phrases. Hence justify each of the complete statements with proper reasons. Provide code snippets where necessary to validate your answers.

i) A ___________block can handle any type of exceptions and it must be placed at _______________.
ii) _________ type conversion is not allowed in _________ blocks.
iii) An exception can be _____________ if it is not fully handled in the _________ block.
iv) A Base class object can be used to handle any exceptions of _________________. However, ____________ class object cannot handle any exceptions of Base class type object.
v) If an object of a class is created within a try block, then on coming out of the try block ,___________ is executed first, and the _______________ is executed next.
vi) Specialized function template is used when ____________________.

b) Consider a function template having the following signature:

```
template <class T=char, int N=3>
void print(T a)
{
for(int i=0;i<N;i++)
 cout<<a<<" ";
}
```

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<int,4> (3);
print<4>('A');
print<double> (5.56);
print<int>(3);
print<>(3.56F);
print<double,5> (4.5);
print<char,4>(5);
print<>();
```

Or,

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

[(2x6)+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.