CMR
INSTITUTE OF
TECHNOLOGY

USN | | | | | | | | | |

Internal Assessment Test – I

SCHEME OF EVALUATION

Sub:	OBJECT ORIENTED CONCEPTS							Code:	15CS45	Marks Distribution
Date:	30/ 03 / 2017	Duration:	90 mins	Max Marks:	50	Sem:	IV	Branch:	CSE (All Sec)	
Answer Any FIVE FULL Questions										

		Marks	OBE CO	OBE RBT	
1 (a)	**Define function Overloading. Write a C++ program with three overloaded function area() to find area of rectangle, area of rectangular box, area of circle.**	[10]	CO1	L3	**Explanation: 4m Program: 6m**

C++ allows two or more functions to have the same name. For this, however, they must have different signatures. *Signature of a function means the number, type, and sequence of formal arguments of the function.* In order to distinguish amongst the functions with the same name, the compiler expects their signatures to be different. Depending upon the type of parameters that are passed to the function call, the compiler decides which of the available definitions will be invoked

// C++ program to find area of rectangular box, circle and rectangle using function overloading.

```
#include<iostream.h>
#include<conio.h>
const float pi=3.14;
float area(float n,float b,float h)
{
        float ar;
        ar=n*b*h;
```

```
        return ar;
}
float area(float r)
{
        float ar;
        ar=pi*r*r;
        return ar;
}
float area(float l,float b)
{
        float ar;
        ar=l*b;
        return ar;
}
void main()
{
        float b,h,r,l;
        float l1,b1,h1;
        float result;
        clrscr();
        cout<<"\nEnter the length  breadth and height of Rectangular box: \n";
        cin>>l1>b1>>h1;
        result=area(,b,h);
        cout<<"\nArea of Rectangular box: "<<result<<endl;
        cout<<"\nEnter the Radius of Circle: \n";
        cin>>r;
        result=area(r);
        cout<<"\nArea of Circle: "<<result<<endl;
        cout<<"\nEnter the Length & Bredth of Rectangle: \n";
        cin>>l>>b;
        result=area(l,b);
        cout<<"\nArea of Rectangle: "<<result<<endl;
}
```

2 (a)	**Why java is known as a platform-neutral language? Elaborate.**	**[05]**	CO1	L2	**Explanation: 5m**

Output of a Java compiler is not executable code. Rather, it is bytecode. Byteco a highly optimized set of instructions designed to be executed by the Java run-time sys which is called the Java Virtual Machine (JVM). Translating a Java program into byte makes it much easier to run a program in a wide variety of environments because only JVM needs to be implemented for each platform. Details of the JVM will differ f platform to platform; all understand the same Java bytecode. If a Java program compiled to native code, then different versions of the same program would have to for each type of CPU connected to the Internet. This is, of course, not a feasible solu Thus, the execution of bytecode by the JVM is the easiest way to create truly port programs. In general, when a program is compiled to an intermediate form and interpreted by a virtual machine, it runs slower than it would run if compiled to execut code. However, with Java, the differential between the two is not so great. Bec bytecode has been highly optimized, the use of bytecode enables the JVM to exe programs much faster than you might expect.

(b)	**Define Type Conversion. What are its Types. Explain with an example**	**[05]**	CO2	L2	**Definition: 2m** **List of types: 1m** **Example : 2m**

In programming its common to assign a value of one type to a variable of another typ the two types are compatible, then Java will perform the conversion automatically. example, it is always possible to assign an int value to a long variable. However, no types are compatible, and thus, not all type conversions are implicitly allowed. For insta there is no automatic conversion defined from double to byte.

Types of conversions :

1. **Java's Automatic Conversions**

2. Casting Incompatible Types

Java's Automatic Conversions

When one type of data is assigned to another type of variable, an *automatic conversion* will take place if the following two conditions are met:

- The two types are compatible.
- The destination type is larger than the source type.

When these two conditions are met, a *widening conversion* takes place. example, the **int** type is always large enough to hold all valid **byte** values, so no exp cast statement is required.

For widening conversions, the numeric types, including integer and floa point types, are compatible with each other. However, there are no automatic convers from the numeric types to **char** or **boolean**. Also, **char** and **boolean** are not compa with each other.

2. Casting Incompatible Types

To create a conversion between two incompatible types, you must use a cast. A cast is simply an explicit type conversion. It has this general form:(target-type) value

Q. No.	Question	Marks	CO	Level	Scheme
3 (a)	**Define Structure. What is a need of it and how it can be used to create new data type. Explain with an example.**	**[10]**	CO2	L2	**Definition: 2m** **Need: 2m** **New datatype:4m** **Example : 2m**

A Structure is a collection of different data types which are grouped together and each element in a *C structure* is called member.

4 (a)	**List and explain the java BUZZWORDS?**	**[07]**	CO1	L2	**List :3** **Explanation:3**

1. Simple
2. Secure
3. Portable
4. Object-oriented
5. Robust
6. Multithreaded
7. Architecture-neutral
8. Interpreted
9. High performance
10. Distributed
11. Dynamic

(b)	**Explain the following OOP features:** **i) Encapsulation ii) Polymorphism**	**[04]**	CO2	L2	**2m each**

Encapsulation in Java is a mechanism of wrapping the data (variables) and code acting on the data (methods) together as a single unit. In encapsulation, the variables of a class will be hidden from other classes, and can be accessed only through the methods of their current class. Therefore, it is also known as **data hiding**.

To achieve encapsulation in Java −

- Declare the variables of a class as private.
- Provide public setter and getter methods to modify and view the variables values.

Example

Following is an example that demonstrates how to achieve Encapsulation in Java −

```
/* File name : EncapTest.java */
public class EncapTest {
   private String name;
   private String idNum;
   private int age;
```

```
    public int getAge() {
        return age;
    }
}
```

Polymorphism:

oops allows two or more functions to have the same name. For this, however, they must have different signatures. *Signature of a function means the number, type, and sequence of formal arguments of the function.* In order to distinguish amongst the functions with the same name, the compiler expects their signatures to be different. Depending upon the type of parameters that are passed to the function call, the compiler decides which of the available definitions will be invoked

Q. No.	Question	Marks	CO	Level	Scheme
5 (a)	**Define Constructor. When it is invoked. Explain with an example.**	[05]	CO2	L2	**Definition :2m** **Invoke : 1m** **Example : 2m**

Definition of Constuctor: The constructor gets called automatically for each object has just got created. It appears as member function of each class, whether it is define not. It has the same name as that of the class. It may or may not take parameters. It doe return anything (not even void).

The prototype of a constructor is

```
<class name> (<parameter list>);
Eg:

class A
{
        int x;
        public:
```

```
        A(); //our own constructor
        void setx(const int=0);
        int getx();
};
/*End of A.h*/
```

(b)	**Explain the following Operators: a) >>> b)short circuit logical operators.**	[05]	CO2	L2	**2.5 each**

A)>>> operator

>>> always shifts zeros into the high-order bit.

The following code fragment demonstrates the >>>. Here, a is set to –1, which sets all 32 bits to 1 in binary. This value is then shifted right 24 bits, filling the top 24 bits with zeros, ignoring normal sign extension. This sets a to 255.

```
int a = -1;
a = a >>> 24;
```

Here is the same operation in binary form to further illustrate what is happening:

11111111 11111111 11111111 11111111 –1 in binary as an int

>>>24

00000000 00000000 00000000 11111111 255 in binary as an int

b)short circuit logical operators

Java provides two interesting Boolean operators not found in many other computer languages Boolean AND and OR operators, and are known as *short-circuit* logical operators. OR operator results in true when A is true, no matter what B is. Similarly, the AND operator results in false when A is false, no matter what B is. If we use the || and && forms, rather than the | and & forms of these operators, Java will not bother to evaluate the right-hand operand when the outcome of the expression can be determined by the left operand alone. This is very useful when the right-hand operand depends on the value of the left one in order to function properly. For example, the following code fragment shows how one can take advantage of short-circuit logical evaluation to be sure that a division operation will be valid before evaluating it:

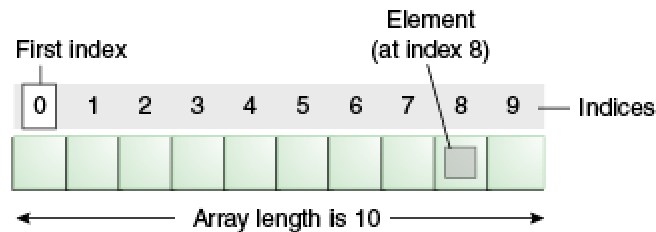
```
if (denom != 0 && num / denom > 10)
```

Q. No.	Question	Marks	CO	RBT	Scheme
6 (a)	**Write a java program to represent planets in the solar system. Each planet has fields for the planets name, its distance from the sun in miles and the number of moons it has. Write a program to read the data for each planet and display.**	[10]	CO2	L3	**Class:2m** **Main():2m** **Read ():2m** **Display():2m** **Proper format :1 m** **Comment: 1m**
7(a)	**Define an Array. Explain with an example "how Array in java differs from Array in C/C++.**	[1	CO2	L2	**Definition :2m** **Java array conc explanation: 4m** **C/C++ array : 4m**

Array is a collection of similar type of elements that have contiguous memory location.

Java array is an object the contains elements of similar data type. It is a data structure where we store similar elements. We can store only fixed set of elements in a java array.

Array in java is index based, first element of the array is stored at 0 index.



In C++, when we declare an array, storage for the array is allocated. In Java, when we declare an array, we really only declare a pointer or reference to an array; storage for the array itself is not allocated until we use the "new" keyword. This difference is elaborated below:

C++

```
int A[10]; // A is an array of length 10
A[0] = 5; // set the 1st element of array A
```

JAVA

```
int [ ] A; // A is a reference / pointer to an array
```

A = new int [10]; // now A points to an array of length 10
A[0] = 5; // set the 1st element of the array pointed to by A

In both C++ and Java we can initialize an array using values in curly braces. Here's the example

Java code:

```
int [ ] myArray = {13, 12, 11}; // myArray points to an array of length 3
// containing the values 13, 12, and 11
```

In Java, a default initial value is assigned to each element of a newly allocated array if no initial value is specified. The default value depends on the type of the array element as shown below:

Type	Value
boolean	false
char	'\u0000'
byte, int, short, long, float, double	0
any reference type e.g. a class object	null

In Java, array bounds are checked and an out-of-bounds array index always causes a runtime error.

8(a)	**List the eight basic data types used in Java. Give examples.**	**[8]**	CO1 L1	**List :4m** **Example : 4m**

Java defines eight primitive types of data**: byte, short, int, long, char, float, double and boolean**. The primitive types are also commonly referred to as simple types

Byte

The smallest integer type is byte. This is a signed 8-bit type that has a range from –128 to 127. Variables of type byte are especially useful when you're working with a stream of data from a network or file. They are also useful when you're working with raw binary data that may not be directly compatible with Java's other built-in types. Byte variables are declared by use of the byte keyword.

For example, the following declares two byte variables called b and c:
byte b, c;

Short

short is a signed 16-bit type. It has a range from –32,768 to 32,767. It is probably the least-used Java type.
Here are some examples of short variable declarations:
short s;
short t;

Int

The most commonly used integer type is int. It is a signed 32-bit type that has a range from –2,147,483,648 to 2,147,483,647. In addition to other uses, variables of type int are commonly employed to control loops and to index arrays
Eg: int a, b;

Long

Long is a signed 64-bit type and is useful for those occasions where an int type is not large enough to hold the desired value. The range of a long is quite large. This makes it useful when big, whole numbers are needed.
Eg: long l;

Char

In Java char is a 16-bit type. The range of a char is 0 to 65,536. There are no negative chars. The standard set of characters known as ASCII still ranges from 0 to 127 as always, and the extended 8-bit character set, ISO-Latin-1, ranges from 0 to 255. Since Java is designed to allow programs to be written for worldwide use, it would use Unicode to represent characters.
Eg: char sex=’f’, male=’m’;

Float

Floating-point numbers, also known as real numbers, are used when evaluating expressions that require fractional precision. For example, calculations such as square root, or transcendental such as sine and cosine, result in a value whose precision

requires a floating-point type. Variables of type float are useful
when you need a fractional component, but don't require a large degree of precision. For example, float can be useful when representing dollars and cents.

Here are some example float variable declarations:

float hightemp, lowtemp;

Double

Double keyword, uses 64 bits to store a value. Manipulating large-valued numbers, double is the best choice.
Eg: double pi, r, a;

Boolean

It can have only one of two possible values, true or false.

boolean b, c = true;
b = false;

Q.No	Question	Marks	CO	Level	Scheme
8(b)	**Define Reference variable. How it is different from pointers?**	**[2]**	CO2	L1	**Definition: 1m** **Difference: 1m**

A reference variable shares the same memory location as the one of which it reference. Therefore, any change in its value automatically changes the value of the vari with which it is sharing memory.

A reference variable is nothing but a reference for an existing variable. It share memory location with an existing variable. The syntax for declaring a reference variable is as follow

```
<data-type> & <ref-var-name>=<existing-var-name>;
```

For example, if 'x' is an existing integer-type variable and we want to declare iRef reference to it the statement is as follows:

```
int & iRef=x;
```

Difference between pointer and references.

- A pointer can be re-assigned any number of times while a reference cannot be re-seated after binding.

- Pointers can point nowhere (`NULL`), whereas reference always refer to an object.
- You can't take the address of a reference like you can with pointers.
    - There's no "reference arithmetics" (but you can take the address of an ol

pointed by a reference and do pointer arithmetics on it as in `&obj + 5`

PO1 - *Engineering knowledge*; PO2 - *Problem analysis*; PO3 - *Design/development of solutions*; PO4 - *Conduct investigations of complex problems*; PO5 - *Modern tool usage*; PO6 - *The Engineer and society*; PO7- *Environment and sustainability*; PO8 – *Ethics*; PO9 - *Individual and team work*; PO10 - *Communication*; PO11 - *Project management and finance*; PO12 - *Life-long learning*