Internal Assessment Test 1 – May 2021

Object Oriented Concepts					Sub Code:	18CS45	Branch:	ISE
21/05/2021	Duration:	90 min's	Max Marks:	50	Sem / Sec:	IV / A, B, C		

Q.No.	Solution and Scheme	Marks
1a.	**Compare and contrast between procedure oriented program and object oriented program**	6M

Ans

Sl. No.		**POP**	**OOP**
	Program Organization	Program is divided into small parts called functions	Program is divided into small parts called objects
	Importance	Importance is not give to data but to functions	Importance is give to data rather than procedures
	Approach	POP follows top down approach	OOP follows bottom up approach
	Access Specifier	Does not have any access specifier	Has three access specifiers namely public, private and protected
	Data Moving	Data can move freely from function to function in the system	Objects can move and communicate with each other
	Maintainability	To add new data and function it is not easy	Provides an easy way to add new data and functions
	Data Access	Function uses global data for sharing that can be accessed freely from function to function in the system	Objects use local data and can be accessed in a control manner.
	Data Hiding	No data hiding is possible, hence security is not possible.	Provides data hiding hence secured programming is possible
	Overloading	Polymorphism is not possible	Polymorphism is possible
	Examples	C, Visual Basic, FORTRAN, Pascal	C++, JAVA, VBNET, C# .NET

Any 6 Correct Points carries 6 marks

1b	**Define 4 pillars of object oriented concepts.**	4M
Ans	**Encapsulation:** Encapsulation is wrapping of data and function or method into a single unit. It is the mechanism that binds together code and data it manipulates, and keeps both safe from outside interference and misuse. Encapsulation is a protective wrapper that prevents code and data from being arbitrarily accessed by other code defined outside the wrapper. Access to the code and data inside the wrapper is tightly controlled through a well defined interface. The power of encapsulated code is that everyone knows how to access it and thus can use it regardless of the implementation details and without fear of unexpected side effects.[1M] **Data Abstraction:** Abstraction means displaying only essential information and hiding the details. Data abstraction refers to providing only essential information about the data to the outside world, hiding the background details or implementation. [1M] **3 Inheritance:** Inheritance is the process by which one object acquires the properties of another object. Inheritance supports the concept of hierarchical classification. For example, a Golden Retriever belongs to the class - **dog**, dog in turn is part of the class **mammal**, and mammal is under the larger class **animal**. Mammal is called the subclass of animals and animals is called the mammal's superclass.[1M] 4 **Polymorphism:** Polymorphism, as the name suggests, is the phenomena by virtue of which the same entity can exist in two or more forms. In OOPS, functions can be made to exhibit polymorphic behaviour. Functions with different set of formal arguments can have the same name. Polymorphism is of two types: static and dynamic[1M]	4M

<table>
<tr><td>2 a.</td><td>Differentiate between class and structure. With an example, explain the syntax for defining, creating and accessing a class and structure.</td><td>5M</td></tr>
<tr><td>Ans</td><td>
<table>
<tr><th>Sl. No.</th><th>CLASS</th><th>STRUCTURE</th></tr>
<tr><td>1</td><td>The members of a class are private by default.</td><td>The members of a structure are public by default.</td></tr>
<tr><td>2</td><td>Classes are of reference types.</td><td>Structs are of value types</td></tr>
<tr><td>3</td><td>A Class can inherit from another class.</td><td>A Struct is not allowed to inherit from another struct or class.</td></tr>
</table>
[1M]
</td><td>5M</td></tr>
<tr><td>Ans</td><td>Example Structure. 2M
struct Distance { // creation of structure
int iFeet;
float fInches;
void setFeet(int x) {
iFeet=x; }
int getFeet() {
return iFeet; }
void setInches(float y) {
fInches=y; }
float getInches() {
return fInches; }
};
int main() {
Distance d1,d2; // creating struct variable
d1.setFeet(2);
d1.setInches(2.2);// accessing functions
d2.setFeet(3);
d2.setInches(3.3);
cout<<d1.getFeet()<<“ ”<<d1.getInches()<<endl;
cout<<d2.getFeet()<<“ ”<<d2.getInches()<<endl;
}
Output
2 2.2
3 3.3</td><td>5M</td></tr>
</table>

	Example Class. 2M
	#include<iostream>
	Using namespace std;
	class Distance // creating class
	{
	int iFeet;
	float fInches;
	public:
	void setFeet(**int**);
	int getFeet();
	}
	void Distance::setFeet(**int** x) { // defining member functions
	iFeet=x; }
	int Distance::getFeet() {
	return iFeet; }
	int main() {
	Distance d1,d2; // creating object of class to access class //members
	d1.setFeet(2);
	cout<<d1.getFeet()<<endl
	}
	
	Note: Any example which explain the syntax for defining, creating and accessing a class and structure.	

2b	Define Inline function. Explain with syntax and example program.	5M

Ans	**An inline function is a function whose compiled code is 'in line' with the rest of the program.** That is, the compiler places a copy of the code of that function at each point where the function is called at compile time. With inline code, the program does not have to jump to another location to execute the code and then jump back. Inline functions, thus, run a little faster than regular functions. [2M]
	
	Example, : use of inline function to return max of two numbers
	Any example code(3Marks)
	#include <iostream>
	using namespace std;
	
	inline int Max(int x, int y)
	{
	return (x > y)? x : y;
	}
	
	// Main function for the program
	int main()
	{
	cout << "Max (20,10): " << Max(20,10) << endl;
	cout << "Max (0,200): " << Max(0,200) << endl;
	cout << "Max (100,1010): " << Max(100,1010) << endl;
	
	}
	OutPut:
	Max (20,10): 20
	Max (0,200): 200
	Max (100,1010): 1010.	

3a	**What is function overloading? Explain with syntax**	

Ans	The feature in C++ which allows two or more functions to have the same name, but with different signatures is called **Function Overloading**.
	Signature of a function means the number, type, and sequence of formal arguments of the function.
	The compiler decides which function is to be called based upon the number, type, and sequence of parameters that are passed to the function.
	Since function prototyping is mandatory in C++, it is possible for the C++ compiler to support function overloading.
	Function overloading enables us to have two functions with the same name and same signature in two different classes.
	
	// Example program to illustrate function overloading using class
	#include<iostream>
	using namespace std;
	
	class A
	{
	Public:
	void show();
	void show(int); //function show() overloaded!!
	};
	
	void A::show()
		3M

	{
	cout<<“Hello\n”;
	}
	void A::show(int x)
	{
	cout << x << endl;
	}
	int main()
	{
	A A1;
	A1.show(); //first definition called
	A1.show(3); //second definition called
	}
	
	Output
	Hello
	3	

3b	**Write a C++ program to find volume of cube (s * s * s), volume of a cylinder (PI * r * r * h), rectangular box (l * b *h) by accepting input from keyboard and printing the volume on console using the method volume() .**	7M

Ans	Program[6M] output [1M]
	#include <iostream>
	
	using namespace std;
	const float pi=3.14;
	
	float vol(float l) //Cube
	{
	return l*l*l;
	}
	float vol(float r,float h) //Cylinder
	{
	return (pi*r*r*h);
	}
	float vol(float l,float b,float h)// rctangular box
	{
	return (l*b*h);
	}
	
	int main()
	{
	float l,r,b,h,t;
	cout<<"\nEnter the Length of Cube: \n";
	cin>>l;
	
	t=vol(l);
	cout<<"\n\nVolume of Cube:"<<t;
	cout<<"\n\nEnter the Radius & Hieght of Cylinder: \n";
	cin>>r>>h;
	t=vol(r,h);
		7M

	cout<<"\n\nVolume of Cylinder: "<<t;
	cout<<"\n\nEnter the Length,Breadth & Hieght of Rectangle: \n";
	cin>>l>>b>>h;
	t=vol(l,b,h);
	cout<<"\n\nVolume of Rectangle: "<<t;
	
	return 0;
	}	

4a	**Explain how one can bridge two classes using friend function. Write a C++ program to find the sum of two numbers using bridge friend function add().**	10M

Ans	Friend function can be used as bridges between two classes. To bridge two classes with a function, the function should be declared as a friend to both the classes. Then the friend function can access private data of both classes. **C++ [2M]**
	
	program to find the sum of two numbers using bridge friend function add()
	
	#include <iostream>
	using namespace std;
	
	class B ; // Forward declaration
	
	class A
	{
	int a;
	public:
	A()
	{
	a = 100;
	cout << "Private member of class A is " << a << endl;
	}
	friend void add(A,B);
	};
	
	class B
	{
	int b;
	public:
	B()
	{
	b = 200;
	cout << "Private member of class B is " << b << endl;
	}
	friend void add(A,B);
		10M

	};
	
	void add (A Aobj, B Bobj)
	{
	cout << "Sum of private members of class A and B = " << (Aobj.a + Bobj.b) << endl;
	}
	
	int main()
	{
	A A1;
	B B1;
	add (A1, B1);
	return 0;
	}
	
	Program with proper syntax 7M
	
	Output : 1M
	Private member of class A is 100
	Private member of class B is 200
	Sum of private members of class A and B = 300	

5a.	**What is reference variable? Explain. Also write a program in C++ to swap two int values and display the values before and after swapping using reference variable and using call by reference.**	10M
Ans	A reference variable can be defined as a reference or alias for an existing variable. It shares the memory location with an existing variable. The syntax for declaring a reference variable is as follows - **<data-type> & <ref-var-name> = < existing-var-name>;** Example: **int & iRef = x;** iRef is a reference to x. This means that although iRef and x have separate entries in the OS, their addresses are actually the same. Thus a change in the value of x will naturally reflect in iRef and vice versa. Reference variables must be initialized at the time of declaration otherwise the compiler will not know what address it has to record for the reference variable. After their creation, **[2M]** **C++ Program to swap two integers using reference variable [Program 3M]** #include <iostream> using namespace std; void swap(int &,int &); int main()	10M

```
{
        int a = 10, b = 20;

        cout << "Before swapping" << endl;
cout << "Value of a is " << a <<endl;
        cout << "Value of b is " << b << endl;
        swap (a,b);

cout << "After Swapping " << endl;
        cout << "Value of a is " << a << endl;
        cout << "Value of b is " << b << endl;
}

void swap (int & a, int & b)
{
        int temp;
        temp = a;
        a = b;

        b = temp;
        cout << "Inside swap function after swapping " << endl;
        cout << "Value of a is " << a << endl;
        cout << "Value of b is  " << b << endl;
}
```

OutPut: 1M

```
Before swapping
Value of a is 10
Value of b is 20
Inside swap function after swapping
Value of a is 20
Value of b is  10
```

Using Call By Reference[Program 3M]

```
#include <iostream>
using namespace std;

void swap(int *xp, int *yp);


int main()
{
   int x , y;
   cout<<"Enter Value of x "<<endl;
   cin>>x;
   cout<<"Enter Value of y "<<endl;
   cin>>y;
   swap(&x, &y);
   cout<<"\nAfter Swapping: x =" << x <<"and"<< "y  is = "<< y<<endl;
   return 0;
}
void swap(int *xp, int *yp)
```

	{
	int temp = *xp;
	*xp = *yp;
	*yp = temp;
	}
	
	Output[1M]
	Enter Value of x
	6
	Enter Value of y
	4
	
	After Swapping: x =4andy is = 6	

6a.	**Write a C++ program to define a class employee having members Emp-id, Emp-name, basic salary and functions accept() and display(). Calculate DA=25% of basic salary, HRA=800, I-tax=15% of basic salary. Display the payslip using appropriate output format.**	10M

Ans	Program Code 8M:
	#include<iostream>
	using namespace std;
	
	class Employee
	{
	int eid;
	char ename[100];
	float basic_salary, hra, da, i_tax, net_salary;
	
	public:
	void accept_details()
	{
	cout<<"\n Enter Employee Id : ";
	cin>>eid;
	cout<<"\n Enter Employee Name : ";
	cin>>ename;
	cout<<"\n Enter Basic Salary : ";
	cin>>basic_salary;
	
	hra = 800;
	da = 0.25 * basic_salary;
	i_tax = 0.15 * basic_salary;
	net_salary = basic_salary + da + hra - i_tax;
	}
	void display_details()
	{
	cout<<"\n ---------- ";
	cout<<"\n Employee Id : "<<eid;
	cout<<"\n Employee Name : "<<ename;
	cout<<"\n Basic Salary : "<<basic_salary;
	cout<<"\n HRA : "<<hra;
	cout<<"\n DA : "<<da;
	cout<<"\n I-Tax : "<<i_tax;
	cout<<"\n Net Salary : "<<net_salary;
	}
	};
	
	int main()
	{
	Employee e;
	e.accept_details();
	e.display_details();
	return 0;
	}
	
	Output: 2M
		10M

```
Enter Employee Id : 1001

Enter Employee Name : Ravi

Enter Basic Salary : 45000

------------------------
Employee Id   : 1001
Employee Name : Ravi
Basic Salary  : 45000
HRA           : 800
DA            : 11250
I-Tax         : 6750
Net Salary    : 50300
```