GOVT. OF KERALA

DEPARTMENT OF TECHNICAL EDUCATION

OFFICE OF THE CONTROLLER OF TECHNICAL EXAMINATIONS

THIRUVANANTHAPURAM

DIPLOMA EXAMINATION IN ENGINEERING /TECHNOLOGY/MANAGEMENT

CORRECTION NOTE - Scheme

Revision & Sub code: (15) 3134

Subject: Object oriented Programming through C++

① Part A

I 5. To convert one data type to another. (1)

Types :-

1. Const_Cast
2. Static_cast
3. dynamic_cast
4. reinterpret_cast

any(1) (1)

② Part B

II (2) One closing braces (}) missing before if (flag == 1) statement.

③ Part C

VII (b) void Test :: operator -()

↑ Spelling mistake.

Scoring Indicators

Code :(15)3134(Object oriented programming through C++)

Qn. No.	Scoring Indicators	Split score		
1	PART-A			
(i)	1.single line comment(//) 2.multiline comment(/*.....*/)	1 mark + 1 mark	2 marks	
(ii)	Function is expanded in line when it is invoked. Prefix the keyword 'inline' along with function definition. Inline float area(float r) {//function body }	Def: 1 mark + Syntax: 1 mark	2 marks 2 marks	
(iii)	Wrapping of data's and functions into a single unit.	Def: 2 marks	2 marks	
(iv)	a) class member access operator(.) b) scope resolution operator(::) c)size of operator(size of) d)conditional operator(?:)	Any two: 2 marks		
(v)	Method of writing single function or class for a family of similar functions or classes in a generic manner.	Def:2 marks		10 marks
II	PART-B			
(1)	Structure is a user defined data type used for packing together data of different types. It is a convenient tool for handling a group of logically related data items.			

	Once the structure is defined, then create variables of that type to access the structure members. Member variables can be accessed using dot or periodic operator. Example: struct student //*structure defenition* { char name[30]; int rollno; float totmark; } struct student S;//*variable declaration* *S.rollno=12678;//member acess.*	Explanation: 3 marks + Example:3 marks	6 marks		
II(2)	void main() { int ar[100],l,I,num,flag=0; cout<<"enter the size of the array"<<endl; cin>>l; for(i=0;i<l;i++) cin>>ar[i]; cout<<"enter the element to be searched:<<endl; cin>>num; for(i=0;i<l;i++) {	Variable declaration : 1 mark + Array read: 1 mark + Search:3 marks + Result display:1 mark	6 marks		

	if(a[i]==num) { flag=1; break; } if(flag==1) cout<<"element found\n"; else cout<<"element not found\n"; }				
2(3)	Scope resolution operator(::) To define function outside class. Syntax returntype class name::function name(arguments) { //function body} Dot operator(.) used to invoke member functions using objects. Syntax object.memberfunction();	Explanation&syntax(::): 3marks + Explanation&syntax(.): 3 marks	6 marks		
2(4)	1.Class 2.Object 3.Data abstraction 4.Data encapsulation 5.Inheritane 6.Polymorphism 7.Dynamic binding 8.Message passing	Explanation of each : 2marks (any three)	6marks		

2(5)	Friend function A non-member function which have the full access rights to the private members of the class . *it is not in the scope of the class to which it is declared as a friend. *it cannot be called using object of that class *it can be invoked like a normal function. *it can be declared in either private or public part of class. class test { …….. public; ……. friend void xyz(); //declaration of friend };	Explanation: 3marks + Example 3marks	6marks		
2(6)	Method of deriving new class from an old one is called inheritance. The old class is called base class(super class) and newly derived class is called derives class or subclass. Syntax for base class class base-class-name { //members of the base class	Explanation: 3marks + Syntax: 3marks	6marks		

2(7)	}; Syntax for defining derived class class derived-class-name:*visibility-mode* base-class-name { //members of derived class }; Visibility mode may be either private or public or protected. Input operator(>>) -it is called extraction operator,is used with standad input stream,cin. -cin treates data stream of characters. -characters flow from cin to the program thorough the input operator. Output operator(<<) -It is called insertion operator. -It is used the standard output stream cout. -cout treats data as a stream of characters. -these characters flow from program to cout through the output operator.	Explanation : 2+2=4marks syntax: 1+1= 2 marks	6marks	30 marks
III(a)	PART-C while loop pretest(entry controlled) First check the condition then the body of the loop will execute			

	Syntax: While(condition is true) { Action1; } Action2; <u>do.....while</u> (Post test or exit - controlled loop) In do.....while at least once the body of the loop will execute. Syntax: do { Action1; }while(condition is true); Action2;	Explanation 2+2=4marks Example 2+2=4	8 marks		
III(b)	1)Arithmetic(+,-,*,/,%) 2)Logical(&&,||,!) 3)Relational(<,>,<=,>==,==,!=) 4)Bitwise(bitwise AND,bitwise OR,bitwise XOR,bitwise left shift,bitwise right shift,bitwise complement).	Listing: 1 mark + Explanation of any 3 (3x2=6)	7marks		
IV(a)	<u>Storage class</u> It defines the visibility and lifetime of a variable within the program.				

IV (b)	1.Automatic 2.Extern 3.Static 4.Register 1.Automatic -it is the default storage class -visibility restricted to the function in which it is declared -lifetime is limited till the time its container function is executing. 2.Extern(global variable) -external variable is declared outside of a function but accessible inside the function block. -visibility spreads all across the program. -lifetime of this is same as the lifetime of a program. 3.Static -visibility of a local variable -lifetime of an external variable -once it is declared inside a function block it doesn't destroyed after the function executed,but retains its value so that it can be used by future function calls. 4.Register -similar in behaviour to automatic variable -register variables are stored in CPU registers so that increases the access speed. void main() {	Listing:2 marks + Explanation of each: 1.5 x4=6	8 marks		

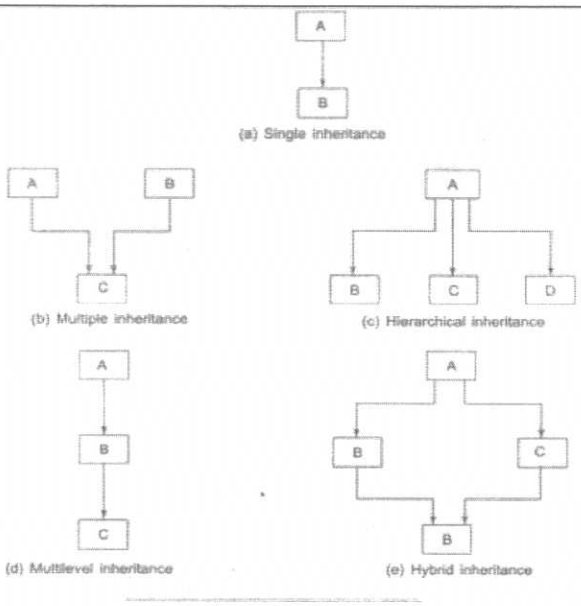
	char st[30],rev[30]; int l,i,j; cout<<"enter the string\n"; cin>>st; l=strlen(st); j=0; For(i=l-1;i>=0;i--) { rev[j]=st[i]; j++; } Cout<<"reversed string="<<rev; }	Char array declaration: 1 mark + Array reading:2 marks + Reverse: 3 marks + Display:1 mark	8marks		
V(a)	<u>Call by reference using pointer</u> Example :swapping two numbers main() { …… …….. swap(&a,&b);//function call …….. }	Pointers: 4 marks + Reference variable :4 marks	8marks		

	void swap(int *p,int *q) //function defenition { int temp; temp=*p; *p=*q; *q=temp; } Call by reference using reference variable main() { …… swap(a,b);//function call …… } void swap(int &p,int &q) //function defenition { int temp; temp=p; p=q; q=temp; }				
V(b)	Function overloading: -usage of same function name to create functions that perform a variety of different tasks.				

VI(a)	-can design a family of functions with one function name but different argument list. -function perform different operations depending on the argument list in the function call. -the correct function to be invoked is determined by checking the number and type of the argument but not on the function type. //function declaration int add(int a,int b)//protoype1 int add(int a,int b,int c);//prototype2 double add(double x,double y);//prototype3 double add(int p,double q); //prototype4 //function call cout<<add(5,10); //use prototype1 cout<<add(15,15.5); //use prototype4 cout<<add(12.5,20.5); //use prototype3 cout<<add(5,20,45); //use prototype2 class Employee { private: char emp_name[40]; int emp_no; float emp_salary; public:	Explanation:4marks + Example: 3marks	7marks		

	void getdata(); void display(); }; void Employee::getdata() { cin>>emp_name>>emp_no>>emp_salary; } void Employee::display() { cout<<emp_name<<emp_no<<emp_salary<<endl; } void main() { Employee E; E.getdata(); E.display(); }	Class declaration:3marks + Member function definition: 3marks + Main : 2marks	8marks		
VI(b)	<u>Constructors</u> -it is a special member function used to initialize the objects of its class. -it is special because it's name is same as that of class name. -it is invoked automatically whenever an object of its associated class is created. -it should be declared in the public section -it doesn't have any return type not even void .				

VII(a)	Example: class A { int m,n; public: A(); //constructor declaration …… …….. } A::A() //constructor defenition { m=0; n=0; } A a1;//object creation 1)Single inheritance 2)Multiple inheritace 3)Herarchical inheritance 4)Multilevel 5)Hybrid	Explanation : 3marks + Example: 4marks	7marks		



Listing:
1 mark
+
Explanation with fig:
4x2=8

9marks

VII(b)

Overloading unary' –' operator

```
class Test
{
    int x,y,z;
Public:
void read(int,int,int);
void operator-();
void display();
};
void Test::read(int a,int b,int c)
{
x=a;
y=b;
z=c;
```

	} void Test::oprator-() { x=-x; y=-y; z=-z; } Void Test::display() { cout<<x<<endl; cout<<y<<endl; cout<<z<<endl; } void main() { Test T; T.read(10,45,60); -T; T.display(); }	Input/output: 2 marks + Overload: 2 marks Main: 2 marks	6marks		
VIII(a)	class Complex { float real,img;				

```
public:
void input_data();
Complex  operator+(Complex);
void print_result();
};
void  Complex::input_data()
{
cout<<"enter real and imaginary part\n";
cin>>real>>img;
}
Complex  Complex::operator+(Complex C)
{
Complex T;
T.real=real+C.real;
T.img=img+C.img;
return(T);
}
void Complex::print_result()
{
cout<<real<<"i"<<img<<endl;
}
main()
{
Complex c1,c2,c3;
c1.input_data();
```

Input/output: 3 marks + Overloading: 3 marks + Main: 3marks

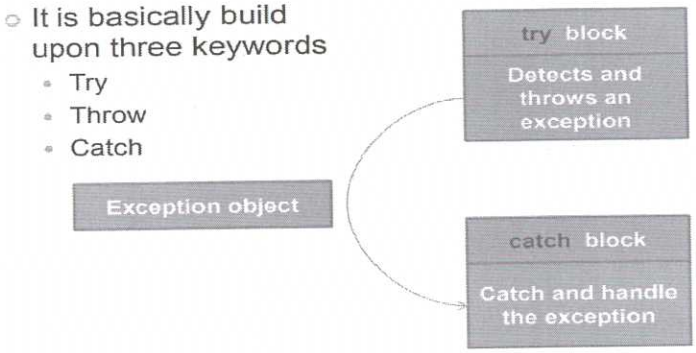
9 marks

VIII(b)	c3=c1+c2; c3.print_result(); } <u>Private inheritance</u>: class derived:private base { //members of derived }; -Public members of the base class become private members of the derived class so that public members of the base class only be accessed by the members of the derived class. -they are not accessed by object of the derived class. -i.e no member of the base class is accessible to the objects of the derived class. <u>Public inheritance</u>: class derived:public base { //members of derived }; -when the class is publicly inherited, public members of the base class become public members of the derived class and can be accessible to the object of the derived class. -private members of the base class are not inheritable to the derived class.	Private inheritance: 3marks +public inheritance: 3marks	6marks	
IX(a)	Polymorphism is achieved through virtual functions. An essential requirement of polymorphism is the ability to refer to objects without regard to their classes.So a single pointer refer to the			

	objects of different classes.Here use the pointer to base class to refer to all the derived class objects.But the base class even it contains the address of the derived class ,always execute the function in the base class.So polymorphism cannot be achieved.This problem can be solved by using virtual function. If the base class and derived class contains functions with the same function name ,the base class function is declared as virtual using keyword ' virtual' preceding its normal declaration. Then c++ determines which function to use at runtime based on the type of the object pointed by the base pointer. class base {…….. public: …….. virtual void show(); }; class derived:public base { Public: …… void show(); …….. }; main() { base b;	Explanation: 2marks+ Program demonstration: 6marks	8marks		

```
derived d;

base *ptr;

ptr=&b;

ptr->show(); //call base class  function

ptr=&d;

ptr->show();//calls derived class function
```

IX(b)



try is used to preface a block of statements which may generate exception. This block of statements are called try block. When an exception is detected, it is throw using a throw statements.

A catch block catches the exception thrown by throw block and handle it appropriately

Explanation: 4marks + Example: 3 marks

7 marks

X(a)

```
try
{
  ..........
  ..............    //block of statements which detects and throw an exceptions
throw exception;
  ..............
  ..............
}
catch(type arg)      //catches exceptions
{
  ..............     // Block of statements that handles the exceptions
  ..............
  ..............
}
```

Class template

When a single class is used to represent a family of similar classes, it is called class template. It avoids the unnecessary repetition of the source code.

Syntax:

```
template <class T>
class classname
{
private:
.....
public:
.....
};
```

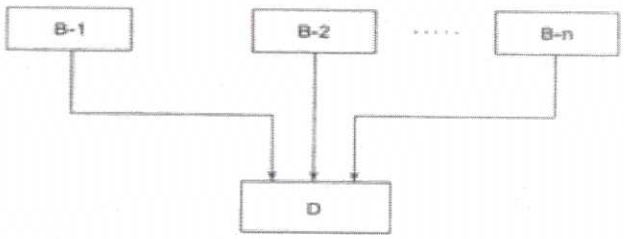
Example:

```
template<class T>     //class declaration
class sample
```

Definition &syntax: 4marks + example program: 4marks

8marks

```
{
private:
T num1,num2;
public:
void getdata();
void display();
};

template <class T>    //member function defenition
void sample<T>::getdata()
{
cin>>num1>>num2;
}
template<class T>
void  sample<T>::display()
{
cout<<num1<<num2<<endl;
}
void main()
{
sample <int> obj1;  //object creation
sample <float> obj2;
obj1.getdata();
```

	obj1.display(); obj2.getdata(); obj2.display(); } Multiple Inheritance:- A class access the attributes of two or more classes .  Fig. 8.8 Multiple inheritance The syntax of a derived class with multiple base classes is as follows: class D: visibility B-1, visibility B-2 ... { ....(Body of D) ..... }; here, *visibility* may be either **public** or **private**. The base classes are separated by commas.	X(b)	Explanation: 4 marks + Example: 3 marks	7marks	

Example:

```
class P : public M, public N
{
  public:
     void display(void);
};
```