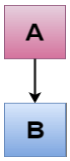
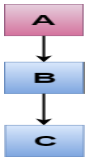
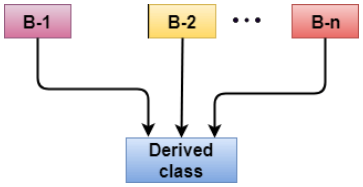
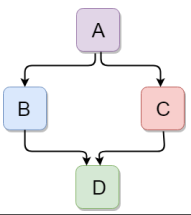
Exam Seat No.:	
End-Sem Examination-II	Winter Year - 2025
Academic Year: 2025-26	Sem: II
Class: F.Y.	Program: B.Tech
Branch Code: ETC	Pattern: 2022
Name of Course: Programming in C++	Course Code: FYE221011
Max. Marks: 60	Duration: 2:30 Hrs.

Answer Key

Q. No.	Details	Max. Marks	CO No.	BT Level

Q.1	What is a Friend Function? Explain with suitable coding example.If a function is defined as a friend function in C++, then the protected and private data of a class can be accessed using the function. By using the keyword friend compiler knows the given function is a friend function. For accessing the data, the declaration of a friend function should be done inside the body of a class starting with the keyword friend. The function can be defined anywhere in the program like a normal C++ function. The function definition does not use either the keyword **friend or scope resolution operator**.
	
	#include<iostrea
	m> using
	namespace std;
	class Test{
	protected:
	int
	a,
	b;
	pu
	bli
	c:
	void setData(int x,int y){
	a=x;b=y;
	}
	friend void getData(Test t);
	};
	void
	getData(Test t){
	cout<<t.a<<"
	"<<t.b;
	}
	int
	main(){
	Test
	t,t1,t2;
	t.setData(
		[6]	CO1	L2

	1,2);
	t1.setData(5,
	6);
	getData(t1);
	return 0;
	}
	(6 marks)			

Q.2	What are different types of inheritance in C++, explain with suitable diagrams. (6 marks) 1. Single inheritance 2. Multiple inheritance 3. Hierarchical inheritance 4. Multilevel inheritance 5. Hybrid inheritance 1. Single inheritance is defined as the inheritance in which a derived class is inherited from the only one base class.  Where 'A' is the base class, and 'B' is the derived class. 2. Multilevel inheritance is a process of deriving a class from another derived class.  3. Multiple inheritance is the process of deriving a new class that inherits the attributes from two or more classes.  4. Hybrid inheritance is a combination of more than one type of inheritance.  	[6]	CO2	L3

	5. Hierarchical inheritance is defined as the process of deriving more than one class from a base class.			

Q.3	**a)** What is function overloading. Explain with suitable example. (8 marks)
	Two or more functions may share the same name but not the same list of arguments when functional overloading occurs. It is a key characteristic of C++. Compile-time polymorphism and function overloading are similar concepts. A close examination reveals that the name stays the same, although the list of arguments, data type, and order all change. The **advantage** of Function overloading is that it increases the readability of the program because you don't need to use different names for the same action.
	
	`#include<iostream>`
	`#include<string>`
	`using namespace std;`
	`class functionOverloadingExample { public:`
	`void myFunc(int a) { cout << "a = " << a << "\n\n"; }`
	`void myFunc(double a) { cout << "a = " << a << "\n\n"; }`
	`void myFunc(int a, int b) { cout << "a = " << a << ", b = " << b << "\n\n"; }`
	`};`
	`int main() { functionOverloadingExample obj; obj.myFunc(10); obj.myFunc(10.20); obj.myFunc(100, 200); return 0; }`
		[16]	CO3	L3

OR

b) What is pointer in C++. List advantages of pointers. (8 marks)

Pointer :- A pointer is a variable that stores the address of another variable. Unlike other variables that helds values of a certain type, pointer holds the address of a variables .

Advantages of Pointers are :-

1. Pointer provide direct access to the memory.
2. Pointer provide a way to returns more than one value to the functions. Pointers provides an alternate way to access array elements.
3. Pointers reduces the storage space and complexity of the program.
4. Pointer reduces the execution of the program

c) Write a C++ program to overload area() function to calculate area of shapes like triangle, square, circle. (8 marks)

```
#include<iostre
m>

using namespace
std; void area(int
a)
{
   int square=a*a;
   cout<<"The Area Of Square Value is:"<<square<<endl;
}
void area(int l,int b)
{
   int rectangle=l*b;
   cout<<" The Area Of The Rectangle Value
is:"<<rectangle<<endl;
}
void area(float r)
{
   float circle=2.34*r*r;
   cout<<"The Area of The Circle Value is:"<<circle<<endl;
}
void area (float le,float br)
{
   float triangle=0.4*le*br;
   cout<<" The Area Of the triangle value
   is:"<<triangle<<endl;
}
int main()
{
```

```
    int
    a,b,l;
    float
    r,le,br;
    cout<<"_____"<<endl;
    cout<<"AREA OF SHAPES USING FUNCTION
 OVERLOADING"<<endl;
    cout<<"___"<<endl;
    cout<<"Enter one value for
    Square:"; cin>>a;
    cout<<endl;
    cout<<"Enter two value for
    Rectangle:"; cin>>b>>l;
    cout<<endl;
    cout<<"Enter one value for
    Circle:"; cin>>r;
    cout<<endl;
    cout<<"Enter two values for triangle:";
    cin>>le>>br;
    cout<<endl;
    cout<<"____
    "<<endl; area(a);
    area(l,b);
    area(r);
    area(le,br);
    cout<<"_____
    "<<endl; return 0;
 }
```

OR

d) Write a C++ program to maintain Employee database using virtual class (8 marks)

```
 #include <iostream>
 #include
 <stdlib.h> using
 namespace std;

 class person
 {
  protected:
      char
      name[20];
      int code;
  public:
      void getdetail(void)
      {
    cout<<"\n\nEnter name
    :- "; cin>>name;
    cout<<"\nEnter code :-
    "; cin>>code;
      }
```

```
    void dispdetail(void)
    {
   cout<<"\n\nNAME      :-
   "<<name; cout<<"\nCODE
                          :-
   "<<code;
    }
};

class account : virtual public person
{
 protected:
     float
 pay;
 public:
     void getpay(void)
     {
    cout<<"\nEnter Pay amount
    :- "; cin>>pay;
     }
     void dispay(void)
     {
    cout<<"\nPAY       :- "<<pay;
     }
};

class admin : virtual public person
{
 protected:
     int
 experience;
 public:
     void getexpr(void)
     {
    cout<<"\nEnter Experience in yrs :- ";
    cin>>experience;
     }
     void dispexpr(void)
     {
    cout<<"\nEXPERIENCE:- "<<experience;
     }
};

class master : public account, public admin
{
   public:
   void create(void)
   {
     cout<<"\n\n=====GETDATA IN=====\n";
     getdetail();
     getpay();
     getexpr();
```

```
    }

    void display(void)
    {
     cout<<"\n\n=====DISPLAY DETAILS=====\n";
     dispdetail();
     dispay();
     dispexpr();
    }

    void update(void)
    {
     cout<<"\n\n=====UPDATE DETAILS=====\n";
     cout<<"\nChoose detail you want to
     update\n"; cout<<"1) NAME\n";
     cout<<"2) CODE\n";
     cout<<"3)
     EXPERIENCE\n";
     cout<<"4) PAY\n";
     cout<<"Enter your choice:-
     "; int choice;
     cin>>choice;
     switch(choic
     e)
     {
       case 1 : cout<<"\n\nEnter name :
            - "; cin>>name;
            break;
       case 2 : cout<<"\n\nEnter code :-
            "; cin>>code;
            break;
       case 3 : cout<<"\n\nEnter pay :-
            "; cin>>pay;
            break;
       case 4 : cout<<"\n\nEnter Expereince :- ";
            cin>>experience;
            break;
       default: cout<<"\n\nInvalid choice\n\n";
     }
    }
};

int main()
{
    master
    ob1;
    int
    choice;
    while(1
    )
    {
```

```
    cout<<"\n\n=====EMPLOYE DATABASE=====\n\n";
    cout<<"\nChoose Operation you want to perform\n";
    cout<<"1) Create Record\n";
    cout<<"2) Display Record\n";
    cout<<"3) Update
    Record\n"; cout<<"4)
    Exit\n"; cout<<"\nEnter
    your choice:- ";
    cin>>choice;
    switch(choice)
    {
     case 1 :
         ob1.create();
         break;
     case 2 :
         ob1.display();
         break;
     case 3 :
         ob1.update();
         break;
     case 4 : exit(1);
     default : cout<<"\n\nInvalid Choice\nTry Again\n\n";
    }
  }
  return 0; }
```

Q.4	**a)** Explain Exception handling in C++. (8 marks) An exception is a problem that arises during the execution of a program. A C++ exception is a response to an exceptional circumstance that arises while a program is running, such as an attempt to divide by zero. Exceptions provide a way to transfer control from one part of a program to another. C++ exception handling is built upon three keywords: try, catch, and throw. ● throw − A program throws an exception when a problem shows up. This is done using a throw keyword. ● catch − A program catches an exception with an exception handler at the place in a program where you want to handle the problem. The catch keyword indicates the catching of an exception. ● try − A try block identifies a block of code for which particular exceptions will be activated. It's followed by one or more catch blocks. Assuming a block will raise an exception, a method catches an exception using a combination of the try and catch keywords. A try/catch exception. Code within a try/catch block is referred to as protected code, and the	[16]	CO4	L3

syntax for using try/catch as follows −

```
try {
  // protected code
} catch( ExceptionName e1 ) {
  // catch block
} catch( ExceptionName e2 ) {
  // catch block
} catch( ExceptionName eN ) {
  // catch block
}
```

tch block is placed around the code that might generate an

OR

b) Develop a C++ program to Create a class of employees (data members - name, DOB, mobile). Develop a function to accept the data and display the information. Use exception handling while accepting the data. e.g in DOB day value should be in between 1 to 31, month value should be in between 1 to 12. (8 marks)

```
#include <iostream>
#include <string>
#include <stdexcept>

using namespace std;

class Employee {
private:
 string name;
 int day, month, year;
 string mobile;

public:
 void acceptData() {
    cout << "Enter employee name: ";

 getline(cin, name);

    // Accept and validate date of birth
    try {
        cout << "Enter DOB - Day: ";
        cin >> day;
        if (day < 1 || day > 31) {
```

```
            throw out_of_range("Day must be between 1 and
31.");
        }

        cout << "Enter DOB - Month: ";
        cin >> month;
        if (month < 1 || month > 12) {
             throw out_of_range("Month must be between 1
and 12.");
        }

        cout << "Enter DOB - Year: ";
        cin >> year;
        if (year < 1900 || year > 2100) {
                throw out_of_range("Year must be realistic
(between 1900 and 2100).");
        }


cin.ignore(); // clear newline character from input buffer
    }
    catch (const out_of_range &e) {
        cerr << "Invalid DOB input: " << e.what() << endl;
        exit(1); // terminate program for simplicity
    }

    cout << "Enter mobile number: ";

getline(cin, mobile);
}

void displayData() {
    cout << "\nEmployee Details:" << endl;
    cout << "Name: " << name << endl;
     cout << "Date of Birth: " << day << "/" << month <<
"/" << year << endl;
    cout << "Mobile: " << mobile << endl;
}
};

int main() {
Employee emp;
emp.acceptData();
emp.displayData();

return 0;
```

```
    }
```

c) What is a template? Explain use of a template with the help of suitable code. (8 marks)

A template is a simple yet very powerful tool in C++. The simple idea is to pass the data type as a parameter so that we don't need to write the same code for different data types. For example, a software company may need to sort() for different data types. Rather than writing and maintaining multiple codes, we can write one sort() and pass the datatype as a parameter.

C++ adds two new keywords to support templates: 'template' and 'type name'. The second keyword can always be replaced by the keyword 'class'.

```
#include
<iostream>
using namespace
std;
template <typename T> T myMax(T x, T y)
{
   return (x > y) ? x : y;
}
 int main()
{
   cout << myMax<int>(3, 7) << endl;
   cout << myMax<double>(3.0, 7.0) <<
   endl; cout << myMax<char>('g', 'e') <<
   endl; return 0;
}
```

OR

d) Write a C++ program to build simple calculator using class template. (8 marks)

```
#include
<iostream>
using namespace
std; template
<class T> class
Calculator
{
private:
      T num1,
num2; public:
      Calculator(T n1, T n2)
      {
            num1
            = n1;
            num2
            = n2;
      }
      void displayResult()
      {
```

```
            cout << "Numbers are: " << num1 << " and " <<
            num2
<< "." << endl;
            cout << "Addition is: " << add() << endl;
            cout << "Subtraction is: " << subtract() <<
            endl; cout << "Product is: " << multiply()
            << endl; cout << "Division is: " << divide()
            << endl;
      }
      T add() { return num1 + num2; }
      T subtract() { return num1 -
      num2; } T multiply() { return
      num1 * num2; } T divide() {
      return num1 / num2; }
};
int main()
{
      Calculator<int> intCalc(2, 1);
      Calculator<float> floatCalc(2.4,
      1.2); cout << "Int results:" <<
      endl; intCalc.displayResult();
      cout << endl << "Float results:" << endl;
      floatCalc.displayResult();
      return 0;
}
```

Q.5

a) Explain different file handling operations with a help of suitable code. (8 marks) [16] CO5 L3

Opening File: fopen() :

We must open a file before it can be read, write, or update. The
fopen() function is used to open a file.

```
#include<stdio.h
> void main( )
{
FILE *fp
; char ch
;
fp =
fopen("file_handle.c","r") ;
while ( 1 )
{
ch = fgetc (
fp ) ; if ( ch
```

```
 == EOF )
 break ;
 printf("%c",c
 h) ;
        }
 fclose (fp ) ;
        }
```

OR

b) Write a simple C++ program to write and read content in the file. (8 marks)

```
 #include<iostrea
 m> using
 namespace std;
 int main()
 {
  ifstream
  in("file1.txt");
  ofstream
  f("file2.txt");
  while(!in.eof())
  {
   string text;
   getline(in,
   text);
   f << text << endl;
  }
  return 0;
 }
```

c) Write a program to count the number of occurrence of a particular character in text file. (8 marks)

```
 #include
 <iostream>
 #include <string>
 using namespace
 std;
  int count(string s, char c)
 {
    int res = 0;
    for (int
      i=0;i<s.length();i++) if
      (s[i] == c)
          re
     s++;
     return
     res;
 }
```

```
int main()
{
   string str=
   "geeksforgeeks"; char c =
   'e';
   cout << count(str, c) <<
   endl; return 0; }
```

OR

d) Write a C++ program to find and replace a specific word in a text file. (8 marks)

```
#include <iostream>
#include <string>
#include
<fstream> using
namespace std;
int main ()
{
string str[1000];
        string header;
        string content = "This is the content area for the
        website"; string footer;
        string
        sidebar;
        string
        source[2];
        int pos[5];
        fstream file;
        size_t
        found;
        file.open
        ("Template.html"); for
        (int i=0; i<1000; i++)
                {
                        getline(file,str[i]);
                }
        for (int i = 0; i<1000; i++)
        {
                        found =
                        str[i].find("header"); if
                        (found!=string::npos)
                        pos[0] = found;
                        found =
                        str[i].find("content"); if
                        (found!=string::npos)
                        pos[1] = found;
                        found =
                        str[i].find("sidebar"); if
                        (found!=string::npos)
                        pos[2] = found;
                        found =
```

```
                        str[i].find("footer"); if
                        (found!=string::npos)
                        pos[3] = found;
                        found =
                        str[i].find("source1"); if
                        (found!=string::npos)
                        pos[5] = found;
                        found =
                        str[i].find("source2"); if
                        (found!=string::npos)
                        pos[6] = found;
        }
        for (int i = 0; i<1000; i++)
                {
                        str[i].replace(pos[1],5,content); <-- My
problem. I randomly entered 5
                }
        for (int i=0; i<1000; i++)
                {
                        cout<<str[i];
            }
  return 0; }
```