An Autonomous College under VTU
DEPARTMENT OF ELECTRONICS AND COMMUNICATION

NAGARJUNA
COLLEGE OF ENGINEERING & TECHNOLOGY

VISION

To transform the students as leaders in Electronics & Communication Engineering to achieve professional excellence in the challenging future

MISSION

M1: To create an environment for the students to have strong academic fundamentals and enable them to be life-long learners.

M2: To provide modern tools to the students in the field of electronics and communication to meet the real-world challenges.

M3: To develop Communication skill, leadership qualities, team work and skills for continuing education the students.

M4: To inculcate Ethics, Human values and skills for solving societal problems and environmental protection.

M5: Validate engineering knowledge through innovative research projects to enhance their employability and entrepreneurship skills.

Program Educational Objectives (PEOs)

PEO-1: Graduates of Electronics and Communication engineering will be using the basic academic knowledge of design and analysis required in the industry for sustainable societal growth.

PEO-2: Graduates of Electronics and Communication engineering will be able to design project based learning and team based learning.

PEO-3: Graduates in Electronics and Communication engineering will demonstrate good communication skills, dynamic leadership qualities with concern for environmental protection.

PEO-4: Electronics and Communication engineering graduates will be capable of pursuing higher studies, take up research and development work blended with ethics and human values.

PEO-5: Electronics and Communication engineering graduates will have the ability to get employed and become entrepreneurs thereby switching over from responsive engineering to creative engineering.

Program Outcomes and Program Specific Outcomes as defined by the Program

Program Outcome:

PO1: Engineering Knowledge: Apply the knowledge of mathematics, science, engineering fundamentals, and electronics and communication engineering principles to the solution of complex problems in electronics and communication engineering.

PO2: Problem Analysis: Identify, formulate, research literature, and analyze complex electronics and communication engineering problems reaching substantiated conclusions using first principles of mathematics, and engineering sciences.

PO3: Design/Development of Solutions: Design solutions for complex electronics and communication engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.

PO4: Conduct Investigations of Complex problems: Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions related to electronics and communication engineering problems.

PO5: Modern Tool Usage: Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modelling to complex electronics and communication engineering activities with an understanding of the limitations.

PO6: The Engineer and Society: Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional electronics and communication engineering practice.

PO7: Environment and Sustainability: Understand the impact of the professional electronics and communication engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.

PO8: Ethics: Apply ethical principles and commit to professional ethics and responsibilities and norms of the electronics and communication engineering practice.

PO9: Individual and Teamwork: Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.

PO10: Communication: Communicate effectively on complex electronics and communication engineering activities with the engineering community and with society at large, such as, being able to comprehend and

write effective reports and design documentation, make effective presentations, and give and receive clearinstructions.

PO11: Project management and finance: Demonstrate knowledge and understanding of the engineering andmanagement principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.

PO12: Life Long Learning: Recognize the need for, and have the preparation and ability to engage inindependent and life-long learning in the broadest context of technological change.

Program Specific Outcome (PSO):

PSO1. Graduate will be able to identify, analyze& solve the problems related to Electronics andCommunication Engineering by applying the fundamental knowledge of Electronics and Communication.

PSO2. Graduate will demonstrate an ability to investigate, design and develop both software and hardwareusing significant knowledge of modern tools in Electronics and Communication Engineering.

PSO3. Graduate will be able to apply their knowledge to assess societal, environmental, health, safetyissues with professional ethics and can also pursue higher studies, involve in research activities, be employable or entrepreneur

IV Semester

C++ Basics			
Course Code	**21EC482**	CIE Marks	50
Teaching Hours/Week (L: T:P: S)	0:0:2:0	SEE Marks	50
Credits	1	Exam Hours	100

Course objectives:

1.Understand object-oriented programming concepts, and apply them in solving problems.
2. To create, debug and run simple C++ programs.
3. Introduce the concepts of functions, friend functions and inline function.
4. Verifying the characteristics of C++
5. Introduce the concepts of exception handling and multithreading.

Sl.No	Experiments
1	Write a C++ program to find largest, smallest & second largest of three numbers using inline functions MAX & Min.
2	Write a C++ program to calculate the volume of different geometric shapes like cube, cylinder and sphere using function overloading concept.
3	Define a STUDENT class with USN, Name & Marks in 3 tests of a subject. Declare an array of 10 STUDENT objects. Using appropriate functions, find the average of the two better marks for each student. Print the USN, Name & the average marks of all the students.
4	Write a C++ program to create class called MATRIX using two-dimensional array of integers, by overloading the operator == which checks the compatibility of two matrices to be added and subtracted. Perform the addition and subtraction by overloading + and – operators respectively. Display the results by overloading the operator <<. If (m1 == m2) then m3 = m1 + m2 and m4 = m1 – m2 else display error
5	Demonstrate simple inheritance concept by creating a base class FATHER with data members: *First Name, Surname, DOB & bank Balance* and creating a derived class SON, which inherits: Surname & Bank Balance feature from base class but provides its own feature: First Name & DOB. Create & initialize F1 & S1 objects with appropriate constructors & display the FATHER & SON details.
6	Write a C++ program to define class name FATHER & SON that holds the income respectively. Calculate & display total income of a family using Friend function.
7	Write a C++ program to initialize the objects using parameter constructor
8	Write a C++ program to explain virtual function (Polymorphism) by creating a base class polygon which has virtual function areas two classes rectangle & triangle derived from polygon & they have area to calculate & return the area of rectangle & triangle respectively.
9	Write a C++ program with different class related through multiple inheritance & demonstrate the use of different access specified by means of members variables & members functions
10	Write a C++ program to implement exception handling with minimum 5 exceptions classes including two built in exceptions

Course outcomes (Course Skill Set):

At the end of the course the student will be able to:

1. Write C++ program to solve simple and complex problems
2. Apply and implement major object-oriented concepts like message passing, function overloading, operator overloading and inheritance to solve real-world problems.
3. Use major C++ features such as Templates for data type independent designs and File I/O to deal with large data set.
4. Analyze, design and develop solutions to real-world problems applying OOP concepts of C++

Assessment Details (both CIE and SEE)

The weightage of Continuous Internal Evaluation (CIE) is 50% and for Semester End Exam (SEE) is 50%. The minimum passing mark for the CIE is 40% of the maximum marks (20 marks). A student shall be deemed to have satisfied the academic requirements and earned the credits allotted to each course. The student has to secure not less than 35% (18 Marks out of 50) in the semester-end examination (SEE).

Continuous Internal Evaluation (CIE):

CIE marks for the practical course is **50 Marks**.

The split-up of CIE marks for record/ journal and test are in the ratio **60:40**.

- Each experiment to be evaluated for conduction with observation sheet and record write-up. Rubrics for the evaluation of the journal/write-up for hardware/software experiments designed by the faculty who is handling the laboratory session and is made known to students at the beginning of the practical session.
- Record should contain all the specified experiments in the syllabus and each experiment write-up will be evaluated for 10 marks.
- Total marks scored by the students are scaled downed to 30 marks (60% of maximum marks).
- Weightage to be given for neatness and submission of record/write-up on time.
- Department shall conduct 02 tests for 100 marks, the first test shall be conducted after the 8th week of the semester and the second test shall be conducted after the 14th week of the semester.
- In each test, test write-up, conduction of experiment, acceptable result, and procedural knowledge will carry a weightage of 60% and the rest 40% for viva-voce.
- The suitable rubrics can be designed to evaluate each student's performance and learning ability. Rubrics suggested in Annexure-II of Regulation book
- The average of 02 tests is scaled down to **20 marks** (40% of the maximum marks).

The Sum of scaled-down marks scored in the report write-up/journal and average marks of two tests is the total CIE marks scored by the student.

Semester End Evaluation (SEE):

SEE marks for the practical course is 50 Marks.

SEE shall be conducted jointly by the two examiners of the same institute, examiners are appointed by the University

All laboratory experiments are to be included for practical examination.

(Rubrics) Breakup of marks and the instructions printed on the cover page of the answer script to be strictly adhered to by the examiners. **OR** based on the course requirement evaluation rubrics shall be decided jointly by examiners.

Students can pick one question (experiment) from the questions lot prepared by the internal /external examiners jointly.

Evaluation of test write-up/ conduction procedure and result/viva will be conducted jointly by examiners.

General rubrics suggested for SEE are mentioned here, writeup-20%, Conduction procedure and result in -60%, Viva-voce 20% of maximum marks. SEE for practical shall be evaluated for 100 marks and scored marks shall be scaled down to 50 marks (however, based on course type, rubrics shall be decided by the examiners)

Change of experiment is allowed only once and 15% Marks allotted to the procedure part to be made zero.

The duration of SEE is 03 hours

Rubrics suggested in Annexure-II of Regulation book

Suggested Learning Resources:

1. Object oriented programming in TURBO C++, Robert Lafore, Galgotia Publications, 2002
2. The Complete Reference C++, Herbert Schildt, 4th Edition, Tata McGraw Hill, 2003.
3. Object Oriented Programming with C++, E Balaguruswamy, 4th Edition, Tata McGraw Hill, 2006.

POS / COs	PO 1	PO 2	PO 3	PO 4	PO 5	PO 6	PO 7	PO 8	PO 9	PO 10	PO 11	PO 12	PSO 1	PSO 2	PSO 3
C216b.1	3	3	3	3	1	-	-	-	3	1	1	1	3	2	1
C216b.2	3	3	3	3	1	-	-	-	3	1	1	1	3	2	1
C216b.3	3	2	3	3	1	-	-	-	3	1	1	1	3	2	1
C216b.4	3	2	3	3	1	-	-	-	3	1	1	1	3	1	1

1). Write a C++ program to find largest, smallest & second largest of three numbers using inline functions max & min.

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

inline int MAXIMUM(int a,int b){
        return(max(a,b));
}
inline int MINIMUM(int a,int b){
        return(min(a,b));
}
int main(){
        int x,y,z,max1,maxf,min1,minf,seclar;
        cout<<"Enter three number: ";
        cin>>x>>y>>z;
        max1=MAXIMUM (x,y);
        maxf=MAXIMUM (max1,z);
        cout<<"The largest value is:
"<<maxf<<"\n";
        min1=MINIMUM(x,y);
        minf=MINIMUM(min1,z);
        cout<<"The smallest value is:
"<<minf<<"\n";
        seclar=(x+y+z)-maxf-minf;
        cout<<"The second largest value is:
"<<seclar;
}
```

```
#include<iostream>
using namespace std;
inline int MAXIMUM(int a,int b,int c){
        return max(max(a,b),c);
}
inline int MINIMUM(int a,int b,int c){
        return min(min(a,b),c);
}
int main(){
        int
x,y,z,max1,maxf,min1,minf,seclar;
        cout<<"Enter three number: ";
        cin>>x>>y>>z;
        max1=MAXIMUM(x,y,z);
        cout<<"The largest value is:
"<<max1<<"\n";
        min1=MINIMUM(x,y,z);
        cout<<"The smallest value is:
"<<min1<<"\n";
        seclar=(x+y+z)-max1-min1;
        cout<<"The second largest value is:
"<<seclar;
}
```

OUTPUT
Enter three number: 12 13 14
The largest value is: 14
The smallest value is: 12
The second largest value is: 13

2). Write a C++ program to calculate the volume of different geometric shapes like cube, cylinder and sphere using function overloading concept.

```
#include<iostream>
using namespace std;
#include <iostream>
using namespace std;
const double pi = 3.14159265359;
// Function to calculate the volume of a cube
double volume(float side) {
   return side * side * side;
}
// Function to calculate the volume of a
cylinder
double volume(double radius, double height) {
   return pi * radius * radius * height;
}
// Function to calculate the volume of a sphere
double volume(double radius) {
   return (4.0 / 3.0) * pi * radius * radius *
radius;
}
int main() {
          float side;
   double radius, height;
   // Calculate the volume of a cube
   cout << "Enter the side length of the cube: ";
   cin >> side;
   cout << "The volume of the cube is: " <<
volume(side) <<"\n";
   // Calculate the volume of a cylinder
   cout << "Enter the radius and height of the
cylinder: ";
   cin >> radius >> height;
   cout << "The volume of the cylinder is: " <<
volume(radius, height) << "\n";
   // Calculate the volume of a sphere
   cout << "Enter the radius of the sphere: ";
   cin >> radius;
   cout << "The volume of the sphere is: " <<
volume(radius) <<"\n";
   return 0;
}
```

Output:

Enter the side length of the cube: 5
The volume of the cube is: 125
Enter the radius and height of the cylinder: 4 5
The volume of the cylinder is: 251.327
Enter the radius of the sphere: 5 23
The volume of the sphere is: 523.599

```
#include<iostream>
using namespace std;
const double pi = 3.14159265359;
// Function to calculate the volume of a cube
int volume(int side) {
   return side * side * side;
}
// Function to calculate the volume of a cylinder
double volume(double radius, double height) {
   return pi * radius * radius * height;
}
// Function to calculate the volume of a sphere
double volume(double radius) {
   return (4.0 / 3.0) * pi * radius * radius * radius;
}
int main() {
          float side;
   double radius, height;
   // Calculate the volume of a cube
   cout << "The volume of the cube is: " << volume(5)
<<"\n";
   // Calculate the volume of a cylinder
   cout << "The volume of the cylinder is: " <<
volume(4.0,5.0) << "\n";
   // Calculate the volume of a sphere
   cout << "The volume of the sphere is: " <<
volume(5.23) <<"\n";
   return 0;
}
```

Output:

The volume of the cube is: 125
The volume of the cylinder is: 251.327
The volume of the sphere is: 599.23

3). **Define a STUDENT class with USN, Name & Marks in 3 tests of a subject. Declare an array of 3 STUDENT objects. Using appropriate functions, find the average of the two better marks for each student. Print the USN, Name & the average marks of all the students.**

```
#include <iostream>
#include <string>
using namespace std;
class STUDENT {
private:
   int usn;
   char name[20];
   int marks[2];
public:
   void read() {
      cout << "Enter USN: ";
      cin >> usn;
      cout << "Enter Name: ";
      cin >> name;
      cout << "Enter Marks in 3 tests: ";
      cin >> marks[0] >> marks[1] >> marks[2];
   }
   double avg() {
      //finding best two marks
      int f= max(max(marks[0],marks[1]),marks[2]);
                  int l= min(min(marks[0],marks[1]),marks[2]);
                  int sec=(marks[0]+marks[1]+marks[2])-f-l;
                  return(f+sec)/2.0;// return the average of the two better marks
   }
   void display() {
      cout << "USN: " << usn << ", Name: " << name << ", Average Marks: " << avg() << endl;
   }
};

int main() {
   STUDENT students[3];
   for (int i = 0; i < 3; i++) {
      cout << "Enter details of student " << i + 1 << endl;
      students[i].read();
   }
   cout << endl << "Details of all students: " << endl;
   for (int i = 0; i < 3; i++) {
      students[i].display();
   }
   return 0;
}
```

OUTPUT:

```
Enter details of student 1
Enter USN: 1
Enter Name: ABC
Enter Marks in 3 tests: 34 44 45
Enter details of student 2
Enter USN: 2
Enter Name: DEF
Enter Marks in 3 tests: 44 43 41
Enter details of student 3
Enter USN: GHI
Enter Name: 45 44 43
Enter Marks in 3 tests: 44 45 43

Details of all students:
USN: 1, Name: ABC, Average Marks: 39
USN: 2, Name: DEF, Average Marks: 43.5
USN: 3, Name: GHI, Average Marks: 44
```

4). Write a C++ program to create class called MATRIX using two-dimensional array of integers, byoverloading the operator == which checks the compatibility of two matrices to be added and subtracted. Perform the addition and subtraction by overloading + and – operators respectively. Display the results by overloading the operator <<. If (m1 == m2) then m3 = m1 + m2 and m4 = m1 – m2 else display error

```
#include <iostream>
using namespace std;
class MATRIX {
  int rows, cols;
  int *mat;
public:
  MATRIX(int r, int c) {
    rows = r;
    cols = c;
    mat = new int*[rows];
    for(int i = 0; i < rows; i++) {
      mat[i] = new int[cols];
    }
  }
MATRIX(const MATRIX &m) {
    rows = m.rows;
    cols = m.cols;
    mat = new int*[rows];
    for(int i = 0; i < rows; i++) {
      mat[i] = new int[cols];
      for(int j = 0; j < cols; j++) {
        mat[i][j] = m.mat[i][j];
      }
    }
  }

  ~MATRIX() {
    for(int i = 0; i < rows; i++) {
      delete[] mat[i];
    }
```

```
    delete[] mat;
}

bool operator ==(const MATRIX &m) const {
    return (rows == m.rows) && (cols == m.cols);
}

MATRIX operator +(const MATRIX &m) const {
    MATRIX result(rows, cols);
    for(int i = 0; i < rows; i++) {
        for(int j = 0; j < cols; j++) {
            result.mat[i][j] = mat[i][j] + m.mat[i][j];
        }
    }
    return result;
}

MATRIX operator -(const MATRIX &m) const {
    MATRIX result(rows, cols);
    for(int i = 0; i < rows; i++) {
        for(int j = 0; j < cols; j++) {
            result.mat[i][j] = mat[i][j] - m.mat[i][j];
        }
    }
    return result;
}

friend ostream& operator <<(ostream &out, const MATRIX &m) {
    for(int i = 0; i < m.rows; i++) {
        for(int j = 0; j < m.cols; j++) {
            out << m.mat[i][j] << " ";
        }
        out << endl;
    }
```

```
        return out;
    }
};

int main() {
    MATRIX m1(2, 2), m2(2, 2);
    m1.mat[0][0] = 1; m1.mat[0][1] = 2;
    m1.mat[1][0] = 3; m1.mat[1][1] = 4;
    m2.mat[0][0] = 5; m2.mat[0][1] = 6;
    m2.mat[1][0] = 7; m2.mat[1][1] = 8;

    if(m1 == m2) {
        MATRIX m3 = m1 + m2;
        MATRIX m4 = m1 - m2;
        cout << "m1 + m2:\n" << m3 << endl;
        cout << "m1 - m2:\n" << m4 << endl;
    }
    else {
        cout << "Error: Incompatible matrices\n";
    }

    return 0;
}
```

5). Demonstrate simple inheritance concept by creating a base class FATHER with data members: *First Name, Surname, DOB & bank Balance* and creating a derived class SON, which inherits: Surname & Bank Balance feature from base class but provides its own feature: First Name & DOB. Create & initialize F1 & S1 objects with appropriate constructors & display the FATHER & SON details.

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

class FATHER {
   protected:
      string first_name;
      string surname;
      string dob;
      int bank_balance;
   public:
      FATHER(string fname, string sname, string d_o_b, int balance) {
       first_name = fname;
       surname = sname;
      dob = d_o_b;
       bank_balance = balance;
      }
void display_details() {
         cout << "Name: " << first_name << " " << surname << endl;
         cout << "DOB: " << dob << endl;
         cout << "Bank Balance: " << bank_balance << endl;
      }
};
class SON : public FATHER {
   public:
      SON(string fname, string sname, string sdob, int balance) : FATHER("", sname, "", balance) {
         first_name = fname;
             dob = sdob;
      }
      void display_details() {
         FATHER::display_details();
         cout << "Relation: SON" << endl;
      }
};

int main() {
   // Create FATHER and SON objects
   FATHER F1("John", "Doe", "01-01-1970", 50000);
   SON S1("David", "Doe", "01-01-2000", 10000);

   // Display details
   cout << "Father's Details:" << endl;
   F1.display_details();
   cout << endl << "Son's Details:" << endl;
   S1.display_details();

   return 0;
}
```

Output:
Father's Details:
Name: John Doe
DOB: 01-01-1970
Bank Balance: 50000

Son's Details:
Name: David Doe
DOB: 01-01-2000
Bank Balance: 10000
Relation: SON

6).Write a C++ program to define class name FATHER & SON that holds the income respectively. Calculate & display total income of a family using Friend function.

```
#include <iostream>
using namespace std;
class SON; // Forward declaration
class FATHER {
private:
   int income;
public:
   FATHER (int income) {
      this->income = income;
   }
// Declare friend function
   friend int calculateTotalIncome(FATHER father, SON son);
};
class SON {
private:
   int income;
public:
   SON(int income) {
      this->income = income;
   }
// Declare friend function
   friend int calculateTotalIncome(FATHER father, SON son);
};
// Define friend function
int calculateTotalIncome(FATHER father, SON son) {
   int totalIncome = father.income + son.income;
   cout << "Total income of the family is " << totalIncome << endl;
   return totalIncome;
}
int main() {
        int fat_inc,son_inc;
        cout<<"Enter father income:";
        cin>>fat_inc;
        cout<<"Enter son income:";
        cin>>son_inc;
   FATHER father(fat_inc);
   SON son(son_inc);
        calculateTotalIncome(father, son);
        return 0;
}
```

output:

Enter father income:45000
Enter son income:35000
Total income of the family is 80000

6). Write a C++ program to initialize the objects using parameter constructor

```
#include <iostream>
#include <string>
using namespace std;
class Person {
public:
  string name;
    int age;
    string occupation;
    // Parameter constructor
    Person(string name1, int age1, string occupation1) {
   name = name1;
  age = age1;
    occupation = occupation;
    }
    void display()
    {
          cout << "Name: " << name << endl;
    cout << "Age: " <<age << endl;
    cout << "Occupation: " << occupation << endl;
          }
};
int main() {
    // Create an object of Person with parameter constructor
    Person john("John Doe", 30, "Software Developer");
    Person peter("peter Doe",35,"Manager");
    // Print the details of the person
    john.display();
    peter.display();
    return 0;
}
```

OUTPUT:

```
Name: John Doe
Age: 30
Occupation: Software Developer
Name: peter Doe
Age: 35
Occupation: Manager
```

8)**Write a C++ program to explain virtual function (Polymorphism) by creating a base class polygon which has virtual function areas two classes rectangle & triangle derived from polygon & they have area to calculate & return the area of rectangle & triangle respectively**

```
#include <iostream>
using namespace std;
class Polygon {
protected:
   int width, height;
public:
   Polygon(int w, int h) {
      width = w;
      height = h;
   }

   // Declare virtual function
   virtual int area() {
      cout << "Area of polygon is not defined" << endl;
      return 0;
   }
};

class Rectangle : public Polygon {
public:
   Rectangle(int w, int h) : Polygon(w, h) {}
```

```
    // Override virtual function
    int area() override {
        return width * height;
    }
};

class Triangle : public Polygon {
public:
    Triangle(int w, int h) : Polygon(w, h) {}

    // Override virtual function
    int area() override {
        return (width * height) / 2;
    }
};

int main() {
    Polygon* poly;

    Rectangle rect(5, 10);
    Triangle tri(8, 6);

    // Assign pointer to Rectangle object and call area() function
    poly = &rect;
    cout << "Area of rectangle is " << poly->area() << endl;

    // Assign pointer to Triangle object and call area() function
    poly = &tri;
    cout << "Area of triangle is " << poly->area() << endl;

    return 0;
}
```

OUTPUT:
Area of rectangle is 50
Area of triangle is 24

9).Write a C++ program with different class related through multiple inheritance & demonstrate the use of different access specified by means of members variables & members functions.

```
#include <iostream>
#include <string>
using namespace std;
class Person {
protected:
    string name;
    int age;
public:
    void set_name(string n) {
        name = n;
    }
    void set_age(int a) {
        age = a;
    }
};

class Student: public Person {
protected:
    int roll_no;
public:
    void set_roll_no(int r) {
        roll_no = r;
    }
```

```
    void display_student() {
        cout << "Name: " << name << endl;
        cout << "Age: " << age << endl;
        cout << "Roll number: " << roll_no << endl;
    }
};

class Employee {
protected:
    int emp_id;
    double salary;
public:
    void set_emp_id(int id) {
        emp_id = id;
    }
    void set_salary(double s) {
        salary = s;
    }
};

class Manager: public Employee, public Person {
protected:
    string department;
public:
    void set_department(string d) {
        department = d;
    }
    void display_manager() {
        cout << "Name: " << name << endl;
        cout << "Age: " << age << endl;
        cout << "Employee ID: " << emp_id << endl;
        cout << "Salary: " << salary << endl;
        cout << "Department: " << department << endl;
    }
};

int main() {
    Student s;
    s.set_name("John");
    s.set_age(20);
    s.set_roll_no(101);
    s.display_student();
    Manager m;
    m.set_name("Jane");
    m.set_age(35);
    m.set_emp_id(1001);
    m.set_salary(50000.0);
    m.set_department("Sales");
    m.display_manager();
    return 0;
}
```

OUTPUT:
Name: John
Age: 20
Roll number: 101
Name: Jane
Age: 35
Employee ID: 1001
Salary: 50000
Department: Sales

10.Write a C++ program to implement exception handling with minimum 5 exceptions classes including two built in exceptions.

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

// Custom exception classes
```

```
class NegativeNumberException : public exception{
public:
   const char* what() const throw(){
      return "Negative number not allowed.";
   }
};

class ZeroDivideException : public exception{
public:
   const char* what() const throw(){
      return "Divide by zero not allowed.";
   }
};

class OutOfRangeException : public exception{
public:
   const char* what() const throw(){
      return "Index out of range.";
   }
};

class InvalidInputException : public exception{
public:
   const char* what() const throw(){
      return "Invalid input.";
   }
};

int main(){
   try{
      int num1, num2, arr[5];
      char op;

      cout<<"Enter two numbers: ";
      cin>>num1>>num2;
      if(num1 < 0 || num2 < 0){
         throw NegativeNumberException();
      }

      cout<<"Enter an operator (+, -, *, /): ";
      cin>>op;
      if(op != '+' && op != '-' && op != '*' && op != '/'){
         throw InvalidInputException();
      }

      int result;
      switch(op){
         case '+': result = num1 + num2; break;
         case '-': result = num1 - num2; break;
         case '*': result = num1 * num2; break;
         case '/':
            if(num2 == 0){
               throw ZeroDivideException();
            }
            result = num1 / num2;
            break;
      }
      cout<<"Result: "<<result<<endl;
```

```
        cout<<"Enter 5 integers: ";
        for(int i=0; i<5; i++){
            cin>>arr[i];
            if(arr[i] < 0 || arr[i] > 100){
                throw OutOfRangeException();
            }
        }
        cout<<"Array: ";
        for(int i=0; i<5; i++){
            cout<<arr[i]<<" ";
        }
        cout<<endl;
    }
    catch(NegativeNumberException e){
        cout<<e.what()<<endl;
    }
    catch(ZeroDivideException e){
        cout<<e.what()<<endl;
    }
    catch(OutOfRangeException e){
        cout<<e.what()<<endl;
    }
    catch(InvalidInputException e){
        cout<<e.what()<<endl;
    }
    catch(exception& e){
        cout<<"Exception: "<<e.what()<<endl;
    }

    return 0;
}
```

OUTPUT:
Enter two numbers: 34 0
Enter an operator (+, -, *, /): /
Divide by zero not allowed.