**Experiment 1: Introduction to C++ and I/O**

**Problem 1:** Write a C++ program to calculate and display the grade of a student based on marks in 5 subjects.

**Code:**

#include <iostream>

using namespace std;

int main() {

    int a,b,c,d,e;

    int sum;

    cout<<"Enter student subjects marks accordingly= ";

    cin>>a>>b>>c>>d>>e;

    sum=(a+b+c+d+e)/5;

    if(sum>=90)

    {

        cout<<"Your grade is A+";

    }

    else if (sum<90 && sum>=80)

    {

        cout<<"Your grade is A";

    }

    else if (sum<80 && sum>=70)

    {

        cout<<"Your grade is B+";

    }

    else if (sum<70 && sum>=60)

    {

        cout<<"Your grade is B";

    }

    else if (sum<60 && sum>=50)

    {

        cout<<"Your grade is C";

    }

    else

    {

        cout<<"Fail";

    }

    return 0;

}

**Output:**

**A screen shot of a computer

AI-generated content may be incorrect.**

**Problem 2:** Write C++ program to find whether number is even or odd.

**Code:**

#include<iostream>

using namespace std;

int main()

{

    int a;

    cout<<"Enter a number =";

    cin>>a;

    if(a%2==0)

    {

        cout<<"The number is even";

    }

    else

    {

        cout<<"The number is odd";

    }

}

**Output:**

A black screen with white text

AI-generated content may be incorrect.

**Experiment 2: Functions and Control Structures**

**Problem 1:** Write a program in C++ using recursive function to compute factorial and check for prime numbers.

**Code:**

#include <iostream>

using namespace std;

int fact=1;

int factorial(int n)

{

    fact=fact\*n;

    n=n-1;

    if (n!=0)

    {

        return factorial(n);

    }

    else

    {

        return fact;

    }

}

int prime(int n)

{

    int count=0;

    for(int i=1;i<=n;i++)

    {

        if(n%i==0)

        {

            count+=1;

        }

    }

    if(count==2)

    {

        cout<<"\nThe number is prime";

    }

    else

    {

        cout<<"\nThe number is not prime";

    }

    return 0;

}

int main()

{

    int n,p;

    cout<<"Enter the number you want factorial for=";

    cin>>n;

    cout<<"Enter the number you want to check if prime=";

    cin>>p;

    int b=factorial(n);

    cout<<"\nFactorial of the function is="<<b;

    prime(p);

    return 0;

}

**Output:**

**A computer screen shot of a computer code

AI-generated content may be incorrect.**

**Problem 2:** Simulate Collatz Conjecture for 1 to N and find the number with longest steps in C++.

**Code:**

#include <iostream>

using namespace std;

void conjecture(int a,int b)

{

    int count1=0;

    while(a!=1)

    {

        if(a%2==0)

        {

            a=a/2;

            count1+=1;

        }

        else

        {

            a=(3\*a+1);

            count1+=1;

        }

    }

    cout<<"Number "<<b<<" "<<":"<<"Steps  "<<count1<<"\n";

}

int main()

{

    int a;

    cout<<"Enter the number =";

    cin>>a;

    cout<<"Collage Simulation upto "<<a<<"\n";

    for(int i=1;i<=a;i++)

    {

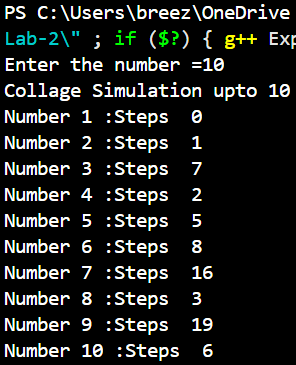
        conjecture(i,i);

    }

    return 0;

}

**Output:**

****

**Problem 3:** Write a C++ program to perform addition and multiplication of Two Matrices.

**Code:**

#include <iostream>

using namespace std;

int main()

{

    int m,n,p,q;

    cout<<"Enter rows and columns of first matrix: ";

    cin>>m>>n;

    cout<<"Enter rows and columns of second matrix: ";

    cin>>p>>q;

    int A[100][100],B[100][100],sum[100][100],product[100][100];

    cout<<"\nEnter elements of first matrix:\n";

    for(int i=0;i<m;i++)

    {

        for(int j=0;j<n;j++)

        {

            cin>>A[i][j];

        }

    }

    cout<<"\nEnter elements of second matrix:\n";

    for(int i=0;i<p;i++)

    {

        for(int j=0;j<q;j++)

        {

            cin>>B[i][j];

        }

    }

    if(m==p && n==q)

    {

        cout<<"\nMatrix Addition:\n";

        for(int i=0;i<m;i++)

        {

            for(int j=0;j<n;j++)

            {

                sum[i][j]=A[i][j]+B[i][j];

                cout<<sum[i][j]<<" ";

            }

            cout<<"\n";

        }

    }

    else

    {

        cout<<"\nMatrix addition is not possible.\n";

    }

    if(n==p)

    {

        for(int i=0;i<m;i++)

        {

            for(int j=0;j<q;j++)

            {

                product[i][j]=0;

            }

        }

        cout<<"\nMatrix Multiplication:\n";

        for(int i=0;i<m;i++)

        {

            for(int j=0;j<q;j++)

            {

                for(int k=0;k<n;k++)

                {

                    product[i][j]+=A[i][k]\*B[k][j];

                }

                cout<<product[i][j]<<" ";

            }

            cout<<"\n";

        }

    }

    else

    {

        cout<<"\nMatrix multiplication is not possible.\n";

    }

    return 0;

}

**Output:**

**A black screen with white text

AI-generated content may be incorrect.**

**Experiment 3: Introduction to Object-Oriented Programming and Encapsulation**

**Problem 1:** Create a class Student with the following:

* Private members: name, rollNumber, marks
* Public methods:
  + setDetails() – to set values
  + displayDetails() – to show student information

**Code:**

#include<iostream>

#include<cstring>

using namespace std;

class Student

{

  private:

    char name[100];

    int rollNumber;

    int marks;

  public:

    void setDetails(char n[],int r,int m)

    {

      strcpy(name,n);

      rollNumber=r;

      marks=m;

    }

    void displayDetails()

    {

      cout<<"Student details\n"<<"Name: "<<name<<"\n"<<"Roll Number: "<<rollNumber<<"\n"<<"Marks: "<<marks;

    }

};

int main()

{

  Student s1;

  char n[100];

  int r;

  int m;

  cout<<"Enter Name: ";

  cin>>n;

  cout<<"Enter Roll Number: ";

  cin>>r;

  cout<<"Enter Marks: ";

  cin>>m;

  s1.setDetails(n,r,m);

  s1.displayDetails();

  return 0;

}

**Output:**

**A screen shot of a computer

AI-generated content may be incorrect.**

**Problem 2:** Implement a Polynomial class with methods to add and evaluate polynomials.

**Code:**

#include<iostream>

using namespace std;

class Polynomial

{

  int sum1=0,sum2=0,sum3=0,eval;

  public:

    void add(int p1[],int p2[])

    {

      sum1=p1[0]+p2[0];

      sum2=p1[1]+p2[1];

      sum3=p1[2]+p2[2];

      cout<<"Sum of Polynomial: "<<sum1<<"x^2+"<<sum2<<"x^1+"<<sum3;

    }

    void evaluate(int x)

    {

      eval=sum1\*(x\*x)+sum2\*(x)+sum3;

      cout<<"\nEvaluation of Sum at x="<<x<<": "<<eval;

    }

};

int main()

{

  Polynomial p;

  int x1,y1,z1,x2,y2,z2,x;

  cout<<"Enter the coefficients accordingly for Polynomial1\n";

  cout<<"Coefficient of x^2= ";

  cin>>x1;

  cout<<"\nCoefficient of x^1= ";

  cin>>y1;

  cout<<"\nCoefficient of x^0=";

  cin>>z1;

  cout<<"Enter the coefficients accordingly for Polynomial2\n";

  cout<<"Coefficient of x^2= ";

  cin>>x2;

  cout<<"\nCoefficient of x^1= ";

  cin>>y2;

  cout<<"\nCoefficient of x^0= ";

  cin>>z2;

  cout<<"Enter the number to be evaluate on= ";

  cin>>x;

  cout<<"Polynomial 1:"<<x1<<"x^2+"<<y1<<"x^1+"<<z1<<"x^0\n";

  cout<<"Polynomial 2:"<<x2<<"x^2+"<<y2<<"x^1+"<<z2<<"x^0\n";

  int p1[]={x1,y1,z1};

  int p2[]={x2,y2,z2};

  p.add(p1,p2);

  p.evaluate(x);

  return 0;

}

**Output:**

**A computer screen shot of a black screen with white text

AI-generated content may be incorrect.**

**Experiment 4: Constructors and Destructors**

**Problem 1:** Write a C++ program to create a Rectangle class.

1. The class should have private data members: length and width (both of type double).
2. Implement a **parameterized constructor** Rectangle(double l, double w) that initializes the length and width of the rectangle.
3. Implement a public method calculateArea() that computes and returns the area of the rectangle (length \* width).
4. Implement a public method displayDimensions() that prints the length and width of the rectangle.

In the main() function, create at least two Rectangle objects using the parameterized constructor with different dimensions, then display their dimensions and calculated areas.

**Code:**

#include<iostream>

using namespace std;

class Rectangle

{

  private:

    double length;

    double width;

  public:

    Rectangle(double l,double w)

    {

      length=l;

      width=w;

    }

    double calculateArea()

    {

      return (length\*width);

    }

    void displayDimensions()

    {

      cout<<"\nThe dimensions of the rectangle ="<<length<<" and "<<width<<"\n";

    }

};

int main()

{

  Rectangle obj1(10,5);

  obj1.displayDimensions();

  cout<<"Area of rectangle = "<<obj1.calculateArea();

  Rectangle obj2(20,10);

  obj2.displayDimensions();

  cout<<"Area of rectangle = "<<obj2.calculateArea();

  return 0;

}

**Output:**

**A screen shot of a computer program

AI-generated content may be incorrect.**

**Problem 2:** Create a DynamicArray class with dynamic allocation and a destructor.

**Code:**

#include<iostream>

using namespace std;

class DynamicArray

{

  private:

    int \*arr;

    int size;

  public:

    DynamicArray(int s)

    {

      size=s;

      arr=new int[size];

      cout<<"DynamicArray constructor called for size "<<size<<"\n";

      for(int i=0;i<size;i++)

      {

        arr[i]=0;

      }

    }

    ~DynamicArray()

    {

      cout<<"DynamicArray destructor called for size "<<size<<"\n";

      delete[] arr;

      arr=NULL;

    }

    void setElement(int i,int value)

    {

      if(i<size)

      {

        arr[i]=value;

      }

    }

    void getElement()

    {

      for(int i=0;i<size;i++)

      {

        cout<<"Element at index "<<i<<" : "<<arr[i]<<"\n";

      }

    }

};

int main()

{

  DynamicArray arr1(5);

  DynamicArray arr2(3);

  cout<<"Array 1\n";

  arr1.setElement(0,10);

  arr1.setElement(1, 20);

  arr1.setElement(4, 50);

  arr1.getElement();

  cout<<"Array 2\n";

  arr2.setElement(0, 100);

  arr2.setElement(2, 300);

  arr2.getElement();

  return 0;

}

**Output:**

**A screenshot of a computer

AI-generated content may be incorrect.**

**Problem 3:** C++ Constructor Delegation with this Pointer

**Code:**

#include <iostream>

#include <string>

using namespace std;

class Product {

    private:

        int id;

        string name;

    public:

        Product(int id, string name){

            cout << "Parameterized constructor called !" << "\n";

            this->id = id;

            this->name = name;

        }

        Product() : Product(0, "Unknown"){

            cout << "Default constructor delegated to parameterized constructor!" << "\n";

        }

        void displayProduct() {

            cout << "Product ID: " << id << ",\nProduct Name: " << name << "\n\n";

        }

};

int main() {

    Product p1;

    p1.displayProduct();

    Product p2(101, "Laptop");

    p2.displayProduct();

    return 0;

}

**Output:**

**A screen shot of a computer

AI-generated content may be incorrect.**

**Experiment-8 Friend function and Static members**

**Problem 1:** Use static variable to count number of objects.

**Code:**

#include <iostream>

using namespace std;

class Widget {

private:

    int id;

    static int objectCount;

    static int nextId;

public:

    Widget() {

        id = ++nextId;

        objectCount++;

        cout << "Widget constructor called. ID: " << id << endl;

    }

    ~Widget() {

        cout << "Widget destructor called. ID: " << id << endl;

        objectCount--;

    }

    static int getObjectCount() {

        return objectCount;

    }

};

int Widget::objectCount = 0;

int Widget::nextId = 0;

int main() {

    cout << "Current Widget count: " << Widget::getObjectCount() << endl;

    Widget w1;

    cout << "Current Widget count: " << Widget::getObjectCount() << endl;

    Widget\* w2 = new Widget();

    cout << "Current Widget count: " << Widget::getObjectCount() << endl;

    {

        Widget w3;

        cout << "Current Widget count: " << Widget::getObjectCount() << endl;

    }

    cout << "Current Widget count after w3 destroyed: " << Widget::getObjectCount() << endl;

    delete w2;

    cout << "Current Widget count after w2 deleted: " << Widget::getObjectCount() << endl;

    return 0;

}

**Output:**

**A screen shot of a computer

AI-generated content may be incorrect.**

**Problem 2:** Write a C++ program to demonstrate the use of a friend function that operates on data from two different classes.

**Code:**

#include <iostream>

using namespace std;

class BankAccount;

class Wallet {

private:

    int cashAmount;

public:

    Wallet(int cash) : cashAmount(cash) {}

    void display() {

        cout << "My Wallet: Cash Amount = $" << cashAmount << endl;

    }

    friend int getTotalFunds(const Wallet& w, const BankAccount& ba);

};

class BankAccount {

private:

    int savings;

public:

    BankAccount(int s) : savings(s) {}

    void display() {

        cout << "My Bank Account: Savings = $" << savings << endl;

    }

    friend int getTotalFunds(const Wallet& w, const BankAccount& ba);

};

int getTotalFunds(const Wallet& w, const BankAccount& ba) {

    return w.cashAmount + ba.savings;

}

int main() {

    Wallet myWallet(500);

    BankAccount myAccount(1500);

    myWallet.display();

    myAccount.display();

    cout << "Total funds (Wallet + BankAccount): $" << getTotalFunds(myWallet, myAccount) << endl;

    return 0;

}

**Output:**

**A computer screen with white text

AI-generated content may be incorrect.**

**Experiment-6 Inheritance**

**Problem 1:** Write a C++ program to demonstrate single inheritance using Person and Student classes.

**Code:**

#include<iostream>

using namespace std;

class Person

{

  private:

    string name;

    int age;

  public:

    Person(string n,int a)

    {

      name=n;

      age=a;

      cout<<"Person constructor called\n";

    }

    void displayPerson()

    {

      cout<<"Person Details:\n"<<" Name: "<<name<<"\n Age: "<<age<<"\n";

    }

};

class Student:public Person

{

  private:

    string StudentId;

    string major;

  public:

    Student(string name,int age,string Id,string maj):Person(name,age)

    {

      StudentId=Id;

      major=maj;

      cout<<"Student constructor called\n";

    }

    void displayStudent()

    {

      displayPerson();

      cout<<"Student Details:\n"<<" StudentID: "<<StudentId<<"\n Major: "<<major<<"\n";

    }

};

int main()

{

  Student student1("Alice Smith", 20, "S1001", "Computer Science");

  student1.displayStudent();

}

**Output:**

**A screen shot of a computer

AI-generated content may be incorrect.**

**Problem 2:** Write a C++ program to illustrate the usage of this pointer and base class member access.

**Code:**

#include <iostream>

#include <string>

using namespace std;

class Vehicle

{

protected:

    string color;

public:

    Vehicle(string c): color(c)

    {

        cout << "Vehicle constructor called\n";

    }

    void displayColor()

    {

        cout << "color:" << color << "\n";

    }

};

class Car : public Vehicle

{

    string model;

    int year;

public:

    Car(string c, string m, int y): Vehicle(c)

    {

        this->model = m;

        this->year = y;

        cout << "Car parameterized constructor called\n";

    }

    Car(): Car("white", "unknown", 2023)

    {

        cout << "Car default constructor called\n";

    }

    void displayCarDetails()

    {

        cout << "car:\n";

        Vehicle::displayColor();

        cout << "model:" << this->model << "\n";

        cout << "year:" << this->year << "\n";

    }

};

int main()

{

    Car c1("white", "crossover", 2024);

    c1.displayCarDetails();

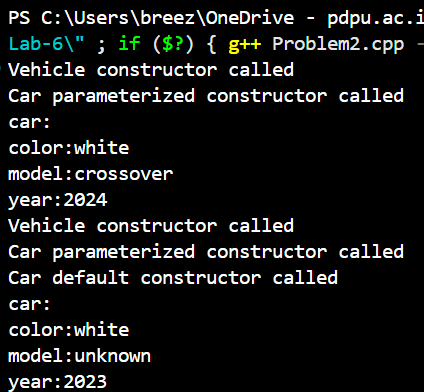
    Car c2;

    c2.displayCarDetails();

    return 0;

}

**Output:**

****