**NAME:- CHANDRA SHEKHARA PRASAD**

**ROLL\_NO:- MCA/19**

// 1) Write a program to create simple calculator for addition, subtraction, division, multiplication,modulus, factorial, gcd, lcm, power, square root, cube root using class and object?

#include <iostream>

#include <cmath>

using namespace std;

class Calculator

{

public:

    void add(int a, int b)

    {

        int sum = a + b;

        cout << "Addition=" << sum << endl;

    }

    void sub(int a, int b)

    {

        int s = a - b;

        cout << "Subtraction=" << s << endl;

    }

    void mult(int a, int b)

    {

        int m = a \* b;

        cout << "Multiplication=" << m << endl;

    }

    void div(int a, int b)

    {

        int d = a / b;

        cout << "Division=" << d << endl;

    }

    void mod(int a, int b)

    {

        int r = a % b;

        cout << "Remainder=" << r << endl;

    }

    /\*int gcd(int a,int b)

    {

        int l=a<b?b:a;

        for(int i=l;i>=1;i--)

        {

            if(a%i==0&&b%i==0)

            {

                return i;

                break;

            }

        }

    }

    int lcm(int a,int b)

    {

        int l=a<b?b:a;

        for(int i=l;i<=a\*b;i++){

            if(i%a==0&&i%b==0)

            {

                return i;

                break;

            }

        }

    }\*/

    void power(int a, int b)

    {

        int p = pow(a, b);

        cout << a << " to the power " << b << " is:" << p << endl;

    }

    void squareRoot(int a)

    {

        int sr = sqrt(a);

        cout << "Square root of " << a << " is:" << sr << endl;

    }

    void cubicRoot(int a)

    {

        int cr = cbrt(a);

        cout << "Cube root of " << a << " is:" << cr << endl;

    }

    int fact(int a)

    {

        if (a == 1)

        {

            return 1;

        }

        else

        {

            return a \* fact(a - 1);

        }

    }

};

int main()

{

    cout << "Hello!!!,i am a digital calculator...." << endl;

    cout << "I have following features...." << endl;

    Calculator calc;

    cout << "1.Addition" << endl;

    cout << "2.Subtraction" << endl;

    cout << "3.Multiplication" << endl;

    cout << "4.Division" << endl;

    cout << "5.Modulus division" << endl;

    cout << "6.Greatest common divisor(GCD)" << endl;

    cout << "7.Least common Multiplier(LCM)" << endl;

    cout << "8.Power" << endl;

    cout << "9.Square root" << endl;

    cout << "10.Cubic root" << endl;

    cout << "11.Factorial" << endl;

    cout << "==============================================" << endl;

    int c;

    string str;

    cout << "Do you want to explore?(Yes/No)" << endl;

    cin >> str;

    while (str == "Yes" || str == "yes")

    {

        cout << "Do you want to continue?(Yes/No)" << endl;

        cin >> str;

        if (str == "Yes" || str == "yes")

        {

            cout << "Enter your choice:";

            cin >> c;

            switch (c)

            {

            case 1:

                cout << "Enter two numbers:";

                int a, b;

                cin >> a >> b;

                calc.add(a, b);

                cout << "=====================" << endl;

                break;

            case 2:

                cout << "Enter two numbers:";

                int d, e;

                cin >> d >> e;

                calc.sub(d, e);

                cout << "=====================" << endl;

                break;

            case 3:

                cout << "Enter two numbers:";

                int f, g;

                cin >> f >> g;

                calc.mult(f, g);

                cout << "=====================" << endl;

                break;

            case 4:

                cout << "Enter two numbers:";

                int h, i;

                cin >> h >> i;

                calc.div(h, i);

                cout << "=====================" << endl;

                break;

            case 5:

                cout << "Enter two numbers:";

                int j, k;

                cin >> j >> k;

                calc.mod(j, k);

                cout << "=====================" << endl;

                break;

            /\*case 6:

            cout<<"Enter two numbers:";

            int l,z;

            cin>>l>>z;

            int hc=calc.gcd(l,z);

            cout<<"GCD of "<<l<<" and "<<z<<" is:"<<hc<<endl;

            cout<<"====================="<<endl;

            break;

            case 7:

            cout<<"Enter two numbers:";

            int n,o;

            cin>>n>>o;

            int lc=calc.lcm(n,o);

            cout<<"LCM of "<<n<<" and "<<o<<" is:"<<lc<<endl;

            cout<<"====================="<<endl;

            break;\*/

            case 8:

                cout << "Enter two numbers:";

                int p, w;

                cin >> p >> w;

                calc.power(p, w);

                cout << "=====================" << endl;

                break;

            case 9:

                cout << "Enter a number:";

                int q;

                cin >> q;

                calc.squareRoot(q);

                cout << "=====================" << endl;

                break;

            case 10:

                cout << "Enter a number:";

                int x;

                cin >> x;

                calc.cubicRoot(x);

                cout << "=====================" << endl;

                break;

            case 11:

                cout << "Enter a number:";

                int y;

                cin >> y;

                int fac = calc.fact(y);

                cout << "Factorial of " << y << " is:" << fac << endl;

                cout << "=====================" << endl;

                break;

                /\*default:

                    cout<<"Enter a valid choice.."<<endl;\*/

            }

        }

        else

        {

            break;

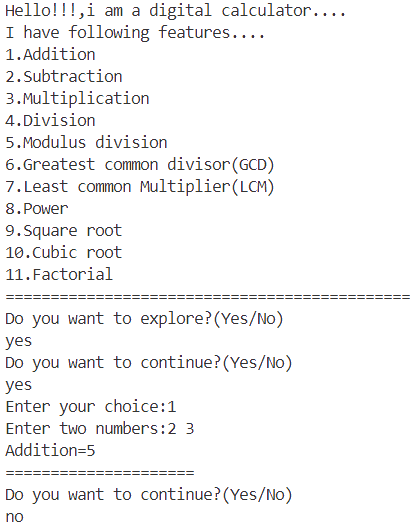
        }

    }

    return 0;

}

**OUTPUT:**



// Write a program to give information about any number such as whether it is even odd, prime or

// not prime, or positive or negative, palindrome or not using class and object?

#include <iostream>

#include <iostream>

using namespace std;

class Prog

{

public:

    int num;

    void positiveornot(int a)

    {

        num = a;

        if (num >= 0)

        {

            cout << num << " is a positive number." << endl;

        }

        else

        {

            cout << num << " is a negative number." << endl;

        }

    }

    void evenornot(int a)

    {

        num = a;

        if (num % 2 == 0)

        {

            cout << num << " is an even number." << endl;

        }

        else

        {

            cout << num << " is an odd number." << endl;

        }

    }

    void primeornot(int a)

    {

        int i;

        num = a;

        for (i = 2; i < num; i++)

        {

            if (num % i == 0)

            {

                break;

            }

        }

        if (num == i)

        {

            cout << num << " is a prime number." << endl;

        }

        else

        {

            cout << num << " is not a prime number." << endl;

        }

    }

    void palindromeornot(int a)

    {

        num = a;

        int temp = num, n, res = 0;

        while (temp > 0)

        {

            n = temp % 10;

            res = res \* 10 + n;

            temp = temp / 10;

        }

        if (res == num)

        {

            cout << num << " is a palindrome number." << endl;

        }

        else

        {

            cout << num << " is not a palindrome number." << endl;

        }

    }

};

int main()

{

    int no;

    cout << "Enter a number:";

    cin >> no;

    Prog numb;

    numb.positiveornot(no);

    numb.evenornot(no);

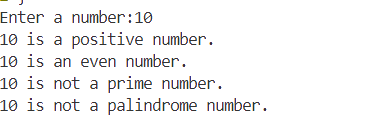
    numb.primeornot(no);

    numb.palindromeornot(no);

    return 0;

}

**OUTPUT:**



// 3) Write a program to find area and perimeter of rectangle using class and object?

#include<iostream>

using namespace std;

class Rectangle{

    public:

        int ln,br;

        int area(int a,int b)

        {

            ln=a,br=b;

            int ar=ln\*br;

            return ar;

        }

        int perimeter(int a,int b)

        {

            ln=a,br=b;

            int pr=2\*(ln+br);

            return pr;

        }

};

int main()

{

    int l,b;

    cout<<"Enter length and breadth of the rectangle:"<<endl;

    cin>>l>>b;

    Rectangle r;

    int a=r.area(l,b);

    cout<<"Area of the rectanhgle having length:"<<l<<" and breadth:"<<b<<" is:"<<a<<" square units."<<endl;

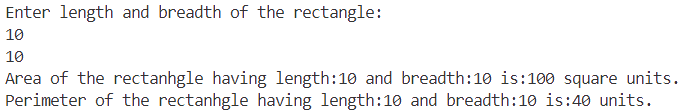
    int p=r.perimeter(l,b);

    cout<<"Perimeter of the rectanhgle having length:"<<l<<" and breadth:"<<b<<" is:"<<p<<" units."<<endl;

    return 0;

}

**OUTPUT:**



// 4) Write a program to find area and perimeter of square using class and object?

#include <iostream>

using namespace std;

class Square

{

public:

    int side;

    int area(int a)

    {

        side = a;

        int ar = side \* side;

        return ar;

    }

    int perimeter(int a)

    {

        side = a;

        int pr = 4 \* side;

        return pr;

    }

};

int main()

{

    int s;

    cout << "Enter side of the square:" << endl;

    cin >> s;

    Square sq;

    int a = sq.area(s);

    cout << "Area of the square having side:" << s << " is:" << a << " square units." << endl;

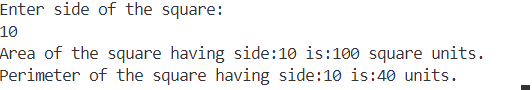
    int p = sq.perimeter(s);

    cout << "Perimeter of the square having side:" << s << " is:" << p << " units." << endl;

    return 0;

}

**OUTPUT:**



// 5) Write a program to find area and perimeter of triangle having 3 sides using class and object?

#include <iostream>

#include <math.h>

using namespace std;

class Triangle

{

public:

    int a, b, c;

    int area(int d, int e, int f)

    {

        a = d, b = e, c = f;

        int s = (a + b + c) / 2;

        int ar = sqrt(s \* (s - 1) \* (s - 2) \* (s - 3));

        return ar;

    }

    int perimeter(int d, int e, int f)

    {

        a = d, b = e, c = f;

        int pr = a + b + c;

        return pr;

    }

};

int main()

{

    int s1, s2, s3;

    cout << "Enter three sides of the triangle:" << endl;

    cin >> s1 >> s2 >> s3;

    Triangle t;

    int a = t.area(s1, s2, s3);

    cout << "Area of the triangle having sides " << s1 << "," << s2 << " and " << s3 << " is:" << a << " square units." << endl;

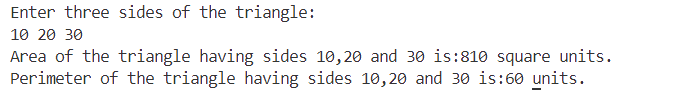
    int p = t.perimeter(s1, s2, s3);

    cout << "Perimeter of the triangle having sides " << s1 << "," << s2 << " and " << s3 << " is:" << p << " units." << endl;

    return 0;

}

**OUTPUT:**



// 6) Write a program to display type of triangle using sides of triangle?

#include <iostream>

using namespace std;

class Triangle

{

public:

    int a, b, c;

    void types(int e, int f, int g)

    {

        a = e, b = f, c = g;

        if (a == b && a == c)

        {

            cout << "This is an Equilateral triangle" << endl;

        }

        else if ((a == b && a != c) || (a == c && a != b) || (b == c && b != a))

        {

            cout << "This is an Isosceles triangle" << endl;

        }

        else

        {

            cout << "This is a scalene triangle" << endl;

        }

    }

};

int main()

{

    int s1, s2, s3;

    cout << "Enter three sides of a triangle:" << endl;

    cin >> s1 >> s2 >> s3;

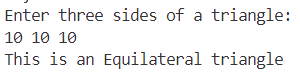
    Triangle t;

    t.types(s1, s2, s3);

    return 0;

}

**OUTPUT:**



// 7) Write a program to create banking application to perform following using using class and

// object?

// a) Initialize with initial balance Rs 3000

// b) Deposit amount if balance is more than 1000 otherwise take Rs 100 as penalty for

// deposit

// c) Withdraw amount if balance is more than 1000 otherwise alert user for low balance

// d) check for balance

#include<iostream>

using namespace std;

class Opt{

    public:

        int bal=3000;

        void balance()

        {

            cout<<"Your current balance is Rs. "<<bal<<endl;

        }

        void deposit()

        {

            if(bal<=1000)

            {

                cout<<"Due to lower balance,you have to pay Rs. 100 as penalty..."<<endl;

            }

            else

            {

                int dep;

                cout<<"Enter the amount you want to deposit:";

                cin>>dep;

                bal=bal+dep;

                cout<<"Rs. "<<dep<<" Successfully deposited."<<endl;

            }

        }

        void withdraw()

        {

            if(bal<=1000)

            {

                cout<<"Alert!!! Low balance....."<<endl;

            }

            else

            {

                int wid;

                cout<<"Enter the amount you want to withdraw:";

                cin>>wid;

                bal=bal-wid;

                cout<<"Rs. "<<wid<<" Successfully withdrawn."<<endl;

            }

        }

};

int main()

{

    cout<<"Welcome to our Bank..."<<endl;

    Opt c;

    cout<<"1. Enter 1 to check balance"<<endl;

    cout<<"2. Enter 2 to deposit cash"<<endl;

    cout<<"3. Enter 3 to withdraw cash"<<endl;

    int a;

    string str;

    cout<<"Do you want to explore our features?(yes/no)";

    cin>>str;

    while(str=="Yes"||str=="yes")

    {

        cout<<"Do you want to continue?(Yes/No)"<<endl;

        cin>>str;

        if(str=="Yes"||str=="yes")

        {

            cout<<"Enter your choice:";

            cin>>a;

            switch(a)

            {

                case 1:

                    c.balance();

                    break;

                case 2:

                    c.deposit();

                    break;

                case 3:

                    c.withdraw();

                    break;

                default:

                    cout<<"Extremely SORRY,may be there is some technical issue!!!! Try again later...."<<endl;

            }

        }

        else

        {

            break;

        }

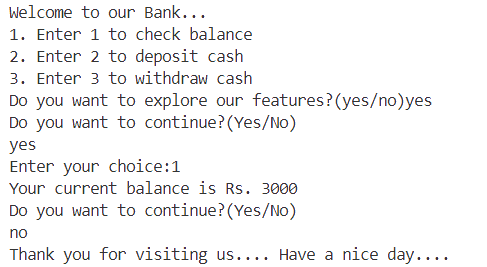
    }

    cout<<"Thank you for visiting us.... Have a nice day...."<<endl;

    return 0;

}

**OUTPUT:**



// 8) Write a program to search for a user defined number in array of 20 numbers using linear search

// and binary search using class and object? Allow user to choose searching method.

#include <iostream>

#include <algorithm>

using namespace std;

class ArraySearch

{

private:

    int arr[20];

    int num;

public:

    int choice;

    void getUserInput()

    {

        cout << "Enter 20 elements for the array:\n";

        for (int i = 0; i < 20; i++)

        {

            cin >> arr[i];

        }

        cout << "Enter the number to search: ";

        cin >> num;

        cout << "Choose search method:\n1. Linear Search\n2. Binary Search\nEnter your choice (1 or 2): ";

        cin >> choice;

    }

    void linearSearch()

    {

        for (int i = 0; i < 20; i++)

        {

            if (arr[i] == num)

            {

                cout << "Number found at index " << i << endl;

                return;

            }

        }

        cout << "Number not found in the array." << endl;

    }

    void binarySearch()

    {

        sort(arr, arr + 20);

        int left = 0, right = 19;

        while (left <= right)

        {

            int mid = left + (right - left) / 2;

            if (arr[mid] == num)

            {

                cout << "Number found at index " << mid << endl;

                return;

            }

            else if (arr[mid] < num)

            {

                left = mid + 1;

            }

            else

            {

                right = mid - 1;

            }

        }

        cout << "Number not found in the array." << endl;

    }

};

int main()

{

    ArraySearch obj;

    obj.getUserInput();

    if (obj.choice == 1)

    {

        obj.linearSearch();

    }

    else if (obj.choice == 2)

    {

        obj.binarySearch();

    }

    else

    {

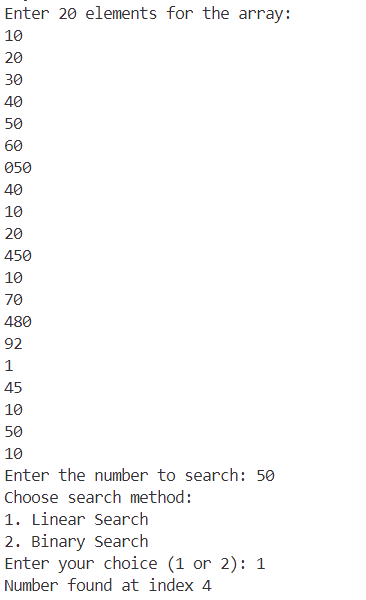
        cout << "Invalid choice." << endl;

    }

    return 0;

}

**OUTPUT:**



// 9) Write a program to implement stack and perform following using class and object?

// a) create an stack of size 10 using array

// b) insert 10 elements into stack using push and isfull method

// c) find factorial of difference between largest and smallest

// element of stack

// d) search any user defined element in stack using peak method

// e) delete 3 elements from stack using pop and isempty method

// f) display remaining element of stack

#include <iostream>

using namespace std;

class Stack

{

private:

    int arr[10];

    int top;

public:

    Stack()

    {

        top = -1;

    }

    bool isEmpty()

    {

        return top == -1;

    }

    bool isFull()

    {

        return top == 9;

    }

    void push(int element)

    {

        if (isFull())

        {

            cout << "Error: Stack is full" << endl;

            return;

        }

        arr[++top] = element;

    }

    int pop()

    {

        if (isEmpty())

        {

            cout << "Error: Stack is empty" << endl;

            return -1;

        }

        return arr[top--];

    }

    int peek(int c)

    {

        if (isEmpty())

        {

            cout << "Error: Stack is empty" << endl;

            return -1;

        }

        for (int i = 0; i < 10; i++)

        {

            if (arr[i] == c)

            {

                return i;

            }

        };

    }

    void display()

    {

        if (isEmpty())

        {

            cout << "Stack is empty" << endl;

            return;

        }

        cout << "Stack elements: ";

        for (int i = top; i >= 0; i--)

        {

            cout << arr[i] << " ";

        }

        cout << endl;

    }

    int fact(int a)

    {

        if (a == 1)

        {

            return 1;

        }

        else

        {

            return a \* fact(a - 1);

        }

    }

};

int main()

{

    Stack stack;

    int arr[10];

    // Insert 10 elements into the stack using push

    cout << "Enter 10 elements to the array:";

    for (int i = 0; i < 10; i++)

    {

        cin >> arr[i];

        stack.push(arr[i]);

    }

    // Display the stack

    stack.display();

    // to find factorial difference of largest and smallest numbers of the stack

    int lg = 0, sm = 10;

    for (int i = 0; i < 10; i++)

    {

        if (arr[i] > lg)

        {

            lg = arr[i];

        }

        if (arr[i] < sm)

        {

            sm = arr[i];

        }

    }

    int diff = lg - sm;

    int fcd = stack.fact(diff);

    cout << "Factorial of diffrence of largest and smallest numbers of the stack is:" << fcd << endl;

    // Search for a user-defined element using peek

    int searchElement;

    cout << "Enter the element to search: ";

    cin >> searchElement;

    int index = stack.peek(searchElement);

    if (index != -1)

    {

        cout << "Element found at index: " << index << endl;

    }

    // Delete 3 elements from the stack using pop

    cout << "Deleting 3 elements from the stack..." << endl;

    for (int i = 0; i < 3; i++)

    {

        stack.pop();

    }

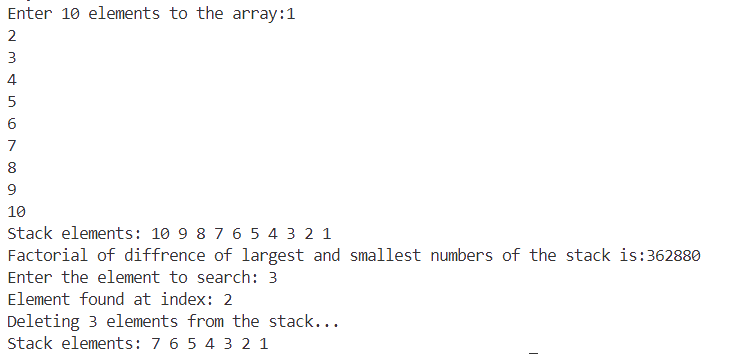
    // Display the remaining elements of the stack

    stack.display();

    return 0;

}

**OUTPUT:**



// 10) Write a program to implement queue and perform following using class and object?

// a) create an queue of size 10 using array

// b) insert 10 elements into queue using insert method

// c) replace even numbers available in queue with nearest prime numbers

// d) display number of odd and prime numbers.

// e) delete 3 elements from queue using delete method

// f) display remaining element of queue.

#include <iostream>

#include <queue>

#include <cmath>

using namespace std;

class Queue

{

private:

    queue<int> q;

public:

    void push(int element)

    {

        q.push(element);

    }

    void pop()

    {

        if (!q.empty())

        {

            q.pop();

        }

    }

    void display()

    {

        if (q.empty())

        {

            cout << "Queue is empty" << endl;

            return;

        }

        cout << "Queue elements: ";

        queue<int> temp = q;

        while (!temp.empty())

        {

            cout << temp.front() << " ";

            temp.pop();

        }

        cout << endl;

    }

    void replaceEvenWithNearestPrime()

    {

        queue<int> temp;

        while (!q.empty())

        {

            int current = q.front();

            q.pop();

            if (current % 2 == 0)

            {

                int nearestPrime = findNearestPrime(current);

                temp.push(nearestPrime);

            }

            else

            {

                temp.push(current);

            }

        }

        q = temp;

    }

    bool isOdd(int num)

    {

        if (num % 2 == 0)

        {

            return false;

        }

        else

        {

            return true;

        }

    }

    bool isPrime(int num)

    {

        if (num <= 1)

        {

            return false;

        }

        for (int i = 2; i <= sqrt(num); i++)

        {

            if (num % i == 0)

            {

                return false;

            }

        }

        return true;

    }

    int findNearestPrime(int num)

    {

        int smaller = num - 1;

        int larger = num + 1;

        while (true)

        {

            if (isPrime(smaller))

            {

                return smaller;

            }

            else if (isPrime(larger))

            {

                return larger;

            }

            smaller--;

            larger++;

        }

    }

};

int main()

{

    Queue queue;

    // Insert 10 elements into the queue using push

    for (int i = 1; i <= 10; i++)

    {

        queue.push(i);

    }

    // Display the queue

    queue.display();

    // to search total number of odds and primes numbers

    int odd = 0, prime = 0;

    for (int i = 1; i <= 10; i++)

    {

        if (queue.isOdd(i))

        {

            odd++;

        }

        if (queue.isPrime(i))

        {

            prime++;

        }

    }

    cout << "Total number of odds:" << odd << endl;

    cout << "Total number of primes:" << prime << endl;

    // Replace even numbers with nearest prime numbers

    queue.replaceEvenWithNearestPrime();

    // Display the modified queue

    cout << "Queue after replacing even numbers with nearest primes: ";

    queue.display();

    // Delete 3 elements from the queue using pop

    cout << "Deleting 3 elements from the queue..." << endl;

    for (int i = 0; i < 3; i++)

    {

        queue.pop();

    }

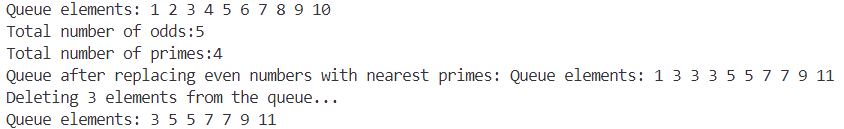
    // Display the remaining elements of the queue

    queue.display();

    return 0;

}

**OUTPUT:**



**Polymorphisim**

// 1) Write a program to find area and perimeter of circle,square,rectangle and triangle using function

// overloading?

#include <iostream>

#include <math.h>

using namespace std;

class Box

{

public:

    void area(float r)

    {

        float ar = (3.14) \* r \* r;

        cout << "Area of the circle is:" << ar << " square units." << endl;

    }

    void area(int s)

    {

        int ar = s \* s;

        cout << "Area of the square is:" << ar << " square units." << endl;

    }

    void area(int ln, int br)

    {

        int ar = ln \* br;

        cout << "Area of the rectangle is:" << ar << " square units." << endl;

    }

    void area(int s1, int s2, int s3)

    {

        int s = (s1 + s2 + s3) / 2;

        int ar = sqrt(s \* (s - s1) \* (s - s2) \* (s - s3));

        cout << "Area of the triangle is:" << ar << " square units." << endl;

    }

    void perimeter(float r)

    {

        float pr = 2 \* (3.14) \* r;

        cout << "Perimeter of the circle is:" << pr << " units." << endl;

    }

    void perimeter(int s)

    {

        int pr = 4 \* s;

        cout << "Perimeter of the square is:" << pr << " units." << endl;

    }

    void perimeter(int ln, int br)

    {

        int pr = 2 \* (ln + br);

        cout << "Perimeter of the rectangle is:" << pr << " units." << endl;

    }

    void perimeter(int s1, int s2, int s3)

    {

        int pr = s1 + s2 + s3;

        cout << "Perimeter of the triangle is:" << pr << " units." << endl;

    }

};

int main()

{

    Box crl;

    float rad;

    cout << "Enter radious of the circle:" << endl;

    cin >> rad;

    crl.area(rad);

    crl.perimeter(rad);

    cout << '\n';

    Box sqr;

    int side;

    cout << "Enter side of the square:" << endl;

    cin >> side;

    sqr.area(side);

    sqr.perimeter(side);

    cout << '\n';

    Box rtnl;

    int l, bd;

    cout << "Enter length and breadth of the rectangle:" << endl;

    cin >> l >> bd;

    rtnl.area(l, bd);

    rtnl.perimeter(l, bd);

    cout << '\n';

    Box trnl;

    int a, b, c;

    cout << "Enter three sides of the triangle:" << endl;

    cin >> a >> b >> c;

    trnl.area(a, b, c);

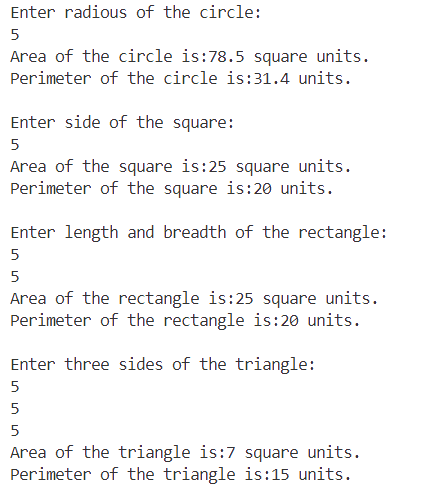
    trnl.perimeter(a, b, c);

    cout << '\n';

    return 0;

}

**OUTPUT:**



// 2) Write program to display following pattern using function overloading?

// Odd number based pyramid starts from value n received from user

//       n+1

//     n+3 n+6

//   n+5 n+10 n+15

// n+7 n+14 n+21 n+28

// even number based pyramid starts from value n received from user

//        n+2

//     n+4 n+6

//   n+6 n+10 n+14

// n+8 n+14 n+20 n+26

// user defined character suppose entered character is k

//    k

//   k k

//  k k k

// k k k k

#include <iostream>

using namespace std;

void displayPattern(char k) {

    cout << "             " << k << endl;

    cout << "         " << k << "       " << k << endl;

    cout << "     " << k << "       " << k << "       " << k << endl;

    cout << " " << k << "       " << k << "      " << k << "       " << k << endl;

}

void printPatternOdd(int n) {

    cout << "Pattern for odd value of n:" << endl;

    cout << "             " << n+1 << endl;

    cout << "         " << n+3 << "       " << n+6 << endl;

    cout << "     " << n+5 << "       " << n+10 << "       " << n+15 << endl;

    cout << " " << n+7 << "       " << n+14 << "      " << n+21 << "       " << n+28 << endl;

    cout << endl;

}

void printPatternEven(int n) {

    cout << "\nPattern for even value of n:" << endl;

    cout << "             " << n+2 << endl;

    cout << "         " << n+4 << "       " << n+8 << endl;

    cout << "     " << n+6 << "       " << n+12 << "       " << n+18 << endl;

    cout << " " << n+8 << "       " << n+16 << "      " << n+24 << "       " << n+32 << endl;

    cout << endl;

}

int main() {

    int oddN, evenN;

    char userChar = 'k';

    displayPattern(userChar);

    cout << endl;

    cout << "Enter an odd value for n: ";

    cin >> oddN;

    if (oddN % 2 == 0) {

        cout << "Please enter an odd value for n." << endl;

        return 1;

    }

    cout << "Enter an even value for n: ";

    cin >> evenN;

    if (evenN % 2 != 0) {

        cout << "Please enter an even value for n." << endl;

        return 1;

    }

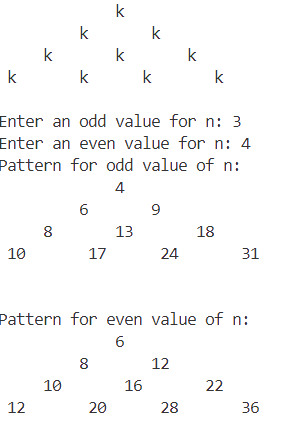
    printPatternOdd(oddN);

    printPatternEven(evenN);

    return 0;

}

**OUTPUT:**



// 3) Write a program to sort 10 numbers in the array using following sorting algorithm and function

// overloading?

// a) merge sort

// b) quick sort

// c) heap sort

#include <iostream>

using namespace std;

class Sort

{

public:

    virtual void sort(int arr[], int n) = 0;

};

class MergeSort : public Sort

{

public:

    void merge(int arr[], int l, int m, int r)

    {

        int i, j, k;

        int n1 = m - l + 1;

        int n2 = r - m;

        int L[n1], R[n2];

        for (i = 0; i < n1; i++)

            L[i] = arr[l + i];

        for (j = 0; j < n2; j++)

            R[j] = arr[m + 1 + j];

        i = 0;

        j = 0;

        k = l;

        while (i < n1 && j < n2)

        {

            if (L[i] <= R[j])

            {

                arr[k] = L[i];

                i++;

            }

            else

            {

                arr[k] = R[j];

                j++;

            }

            k++;

        }

        while (i < n1)

        {

            arr[k] = L[i];

            i++;

            k++;

        }

        while (j < n2)

        {

            arr[k] = R[j];

            j++;

            k++;

        }

    }

    void sort(int arr[], int n) override

    {

        int currSize;

        int leftStart;

        for (currSize = 1; currSize <= n - 1; currSize = 2 \* currSize)

        {

            for (leftStart = 0; leftStart < n - 1; leftStart += 2 \* currSize)

            {

                int mid = min(leftStart + currSize - 1, n - 1);

                int rightEnd = min(leftStart + 2 \* currSize - 1, n - 1);

                merge(arr, leftStart, mid, rightEnd);

            }

        }

    }

};

class QuickSort : public Sort

{

public:

    int partition(int arr[], int low, int high)

    {

        int pivot = arr[high];

        int i = (low - 1);

        for (int j = low; j <= high - 1; j++)

        {

            if (arr[j] < pivot)

            {

                i++;

                swap(arr[i], arr[j]);

            }

        }

        swap(arr[i + 1], arr[high]);

        return (i + 1);

    }

    void sort(int arr[], int n) override

    {

        quickSortUtil(arr, 0, n - 1);

    }

private:

    void quickSortUtil(int arr[], int low, int high)

    {

        if (low < high)

        {

            int pi = partition(arr, low, high);

            quickSortUtil(arr, low, pi - 1);

            quickSortUtil(arr, pi + 1, high);

        }

    }

};

class HeapSort : public Sort

{

public:

    void heapify(int arr[], int n, int i)

    {

        int largest = i;

        int left = 2 \* i + 1;

        int right = 2 \* i + 2;

        if (left < n && arr[left] > arr[largest])

            largest = left;

        if (right < n && arr[right] > arr[largest])

            largest = right;

        if (largest != i)

        {

            swap(arr[i], arr[largest]);

            heapify(arr, n, largest);

        }

    }

    void sort(int arr[], int n) override

    {

        for (int i = n / 2 - 1; i >= 0; i--)

            heapify(arr, n, i);

        for (int i = n - 1; i >= 0; i--)

        {

            swap(arr[0], arr[i]);

            heapify(arr, i, 0);

        }

    }

};

void printArray(int arr[], int n)

{

    for (int i = 0; i < n; i++)

        cout << arr[i] << " ";

    cout << endl;

}

int main()

{

    int arr[10];

    cout << "Enter 10 elements: ";

    for (int i = 0; i < 10; i++)

    {

        cin >> arr[i];

    }

    cout << "array[10]:";

    for (int i = 0; i < 10; i++)

    {

        cout << arr[i] << " ";

    }

    cout << endl;

    int choice;

    cout << "Enter your choice (1 for Merge Sort, 2 for Quick Sort, 3 for Heap Sort): ";

    cin >> choice;

    Sort \*sort;

    switch (choice)

    {

    case 1:

        sort = new MergeSort();

        break;

    case 2:

        sort = new QuickSort();

        break;

    case 3:

        sort = new HeapSort();

        break;

    default:

        cout << "Invalid choice." << endl;

        return 1;

    }

    sort->sort(arr, 10);

    cout << "Sorted array: ";

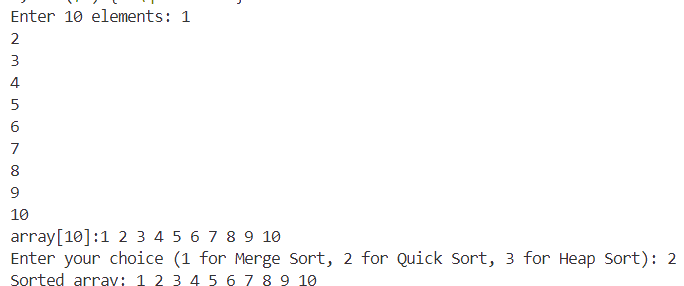
    printArray(arr, 10);

    delete sort;

    return 0;

}

**OUTPUT:**



// 4) Write a program to calculate interest that may be simple or compound using function overloading?

#include <iostream>

#include <math.h>

using namespace std;

class Calc

{

public:

    int p, t, n;

    float r;

    void interest(int p, int t, float r)

    {

        float si = (float)(p \* t \* r) / 100.0;

        cout << "Simple interest=" << si << endl;

    }

    void interest(int p, int t, float r, int n)

    {

        float a = p \* (pow(1 + (r / n), (n \* t)));

        float ci = a - p;

        cout << "Compound interest=" << ci << endl;

    }

};

int main()

{

    int p, t, n;

    float r;

    cout << "Enter principal amount,rate of interest,time:" << endl;

    cin >> p >> r >> t;

    cout << "Enter the number of times interest is compounded per year:" << endl;

    cin >> n;

    Calc a;

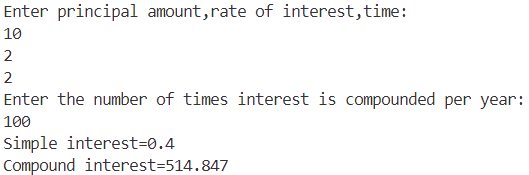
    a.interest(p, t, r);

    a.interest(p, t, r, n);

    return 0;

}

**OUTPUT:**



// 5) Write a program to overload sum method if arguments are numbers than it will add numbers or

// string than concat two strings using function overloading?

#include<iostream>

using namespace std;

class Ops{

    public:

        int sum(int a,int b)

        {

            return (a+b);

        }

        string sum(string a,string b)

        {

            return (a+b);

        }

};

int main(){

    Ops o;

    cout<<"Enter two integers:"<<endl;

    int a,b;

    cin>>a>>b;

    int c=o.sum(a,b);

    cout<<a<<"+"<<b<<"="<<c<<endl;

    cout<<"Enter two strings(first name and last name):"<<endl;

    string f,l;

    cin>>f>>l;

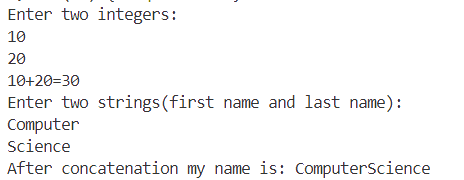
    string h=o.sum(f,l);

    cout<<"After concatenation my name is: "<<h<<endl;

    return 0;

}

**OUTPUT:**

****

// 6) Write a program to check whether a number or string is palindrome or not using function

// overloading?

#include <iostream>

#include <string.h>

using namespace std;

class A

{

public:

    void pallindrome(int num)

    {

        int temp = num, n, res = 0;

        while (temp > 0)

        {

            n = temp % 10;

            res = res \* 10 + n;

            temp = temp / 10;

        }

        if (res == num)

        {

            cout << num << " is a pallindrome number." << endl;

        }

        else

        {

            cout << num << " is not a pallindrome number." << endl;

            ;

        }

    }

    int pallindrome(string str)

    {

        string s = str;

        for (int i = 0; i < s.length() / 2; i++)

        {

            if (s[i] == s[s.length() - i - 1])

            {

                return 1;

            }

            else

            {

                return 0;

            }

        }

    }

};

int main()

{

    A a;

    int num;

    cout << "Enter a number:" << endl;

    cin >> num;

    a.pallindrome(num);

    string str;

    cout << "Enter a string:" << endl;

    cin >> str;

    int st = a.pallindrome(str);

    if (st == 1)

    {

        cout << str << " is a pallindrome." << endl;

    }

    else

    {

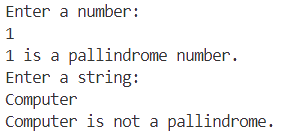
        cout << str << " is not a pallindrome." << endl;

    }

    return 0;

}

**OUTPUT:**

****

// 7) Write a program to find volume of rectangle and square box using function overloading?

#include <iostream>

#include <math.h>

using namespace std;

class Box

{

public:

    void volume(int s)

    {

        int v = s \* s \* s;

        cout << "Volume of the square is:" << v << " cubic units." << endl;

    }

    void volume(int ln, int br, int ht)

    {

        int v = ln \* br \* ht;

        cout << "Volume of the rectangle is:" << v << " cubic units." << endl;

    }

};

int main()

{

    Box sqr;

    int side;

    cout << "Enter side of the square:" << endl;

    cin >> side;

    sqr.volume(side);

    cout << '\n';

    Box rtnl;

    int l, bd, h;

    cout << "Enter length,breadth and height of the rectangle:" << endl;

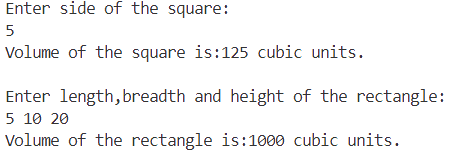
    cin >> l >> bd >> h;

    rtnl.volume(l, bd, h);

    return 0;

}

**OUTPUT:**

****

// 8) Write a program to add two complex numbers using operator overloading?

#include <iostream>

#include <math.h>

using namespace std;

class Complex

{

public:

    int a, b;

    Complex(int rp = 0, int ip = 0)

    {

        a = rp, b = ip;

    }

    void sum()

    {

        if (b >= 0)

        {

            cout << "The complex number is:" << a << "+" << b << "i" << endl;

        }

        else

        {

            cout << "The complex number is:" << a << "-" << abs(b) << "i" << endl;

        }

    }

    Complex operator+(Complex &o)

    {

        Complex t;

        t.a = a + o.a;

        t.b = b + o.b;

        return t;

    }

};

int main()

{

    int rp1, ip1;

    cout << "Enter real and imaginary part of 1st number:" << endl;

    cin >> rp1 >> ip1;

    Complex n1(rp1, ip1);

    n1.sum();

    int rp2, ip2;

    cout << "Enter real and imaginary part of 2nd number:" << endl;

    cin >> rp2 >> ip2;

    Complex n2(rp2, ip2);

    n2.sum();

    Complex n3 = n1 + n2;

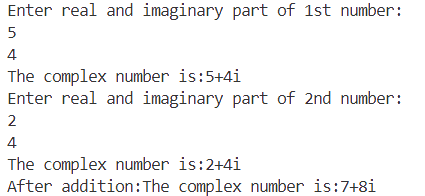
    cout << "After addition:";

    n3.sum();

    return 0;

}

**OUTPUT:**

****

// 9) Write a program to find area and perimeter of 3 boxes in which dimension of box is known to user

// and the 2nd box is 3/4th of 1st box and third box is the addition of 1st and 2nd box using operator

// overloading?

#include <iostream>

using namespace std;

class Box

{

public:

    float ln, br;

    Box(float a = 0, float b = 0)

    {

        ln = a;

        br = b;

    }

    void area()

    {

        float ar = ln \* br;

        cout << "Area of the box is:" << ar << endl;

    }

    void perimeter()

    {

        float pr = 2 \* (ln + br);

        cout << "Perimeter of the box is:" << pr << endl;

    }

    Box operator+(Box &o)

    {

        Box t;

        t.ln = ln + o.ln;

        t.br = br + o.br;

        return t;

    }

};

int main()

{

    float a, b;

    cout << "Enter length and breadth of the box1:" << endl;

    cin >> a >> b;

    cout << "Box1..................." << endl;

    Box b1(a, b);

    b1.area();

    b1.perimeter();

    cout << "Box2..................." << endl;

    float d = a \* 3 / 4.0;

    float e = b \* 3 / 4.0;

    Box b2(d, e);

    b2.area();

    b2.perimeter();

    cout << "Box3..................." << endl;

    Box b3 = b1 + b2;

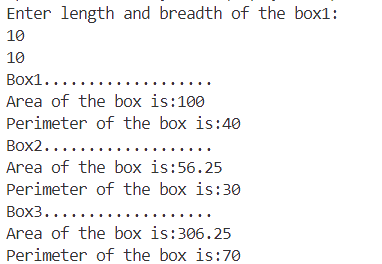
    b3.area();

    b3.perimeter();

    return 0;

}

**OUTPUT:**

****

**Inheritance**

// 1. WAP to find area and volume of a TV using simple inheritance?

#include <iostream>

using namespace std;

class TV

{

public:

    int ln, br;

    TV(int a, int b)

    {

        ln = a, br = b;

    }

    void area()

    {

        int v = ln \* br;

        cout << "Area of the TV is " << v << " square units." << endl;

    }

};

class T : public TV

{

public:

    int ht;

    T(int a, int b, int c) : TV(a, b)

    {

        ht = c;

    }

    void volume()

    {

        int v = ln \* br \* ht;

        cout << "Volume of the TV is " << v << " cubic units." << endl;

    }

};

int main()

{

    int a, b, c;

    cout << "Enter length,breadth and height of the TV:" << endl;

    cin >> a >> b >> c;

    T d(a, b, c);

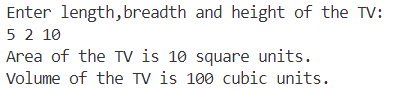
    d.area();

    d.volume();

    return 0;

}

**OUTPUT:**



// 2. WAP to find area, volume and weight of a box using multilevel inheritance?

#include <iostream>

using namespace std;

class Box1

{

public:

    int ln, br;

    Box1(int a, int b)

    {

        ln = a, br = b;

    }

    int area()

    {

        int ar = ln \* br;

        return ar;

    }

};

class Box2 : public Box1

{

public:

    int ht;

    Box2(int a, int b, int c) : Box1(a, b)

    {

        ht = c;

    }

    int volume()

    {

        int ar = area();

        int v = ar \* ht;

        return v;

    }

};

class Box3 : public Box2

{

public:

    int wpv;

    Box3(int a, int b, int c, int d) : Box2(a, b, c)

    {

        wpv = d;

    }

    int weight()

    {

        int v = volume();

        int w = wpv \* v;

        return w;

    }

};

int main()

{

    int a, b, c, d;

    cout << "Enter length,breadth and height of the box:" << endl;

    cin >> a >> b >> c;

    cout << "Enter weight per volume of the box:" << endl;

    cin >> d;

    Box3 b1(a, b, c, d);

    int ar = b1.area();

    int v = b1.volume();

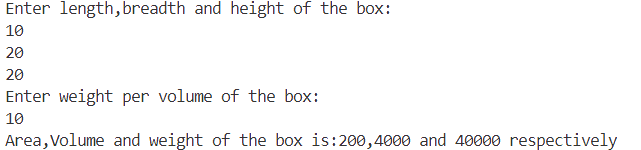
    int w = b1.weight();

    cout << "Area,Volume and weight of the box is:" << ar << "," << v << " and " << w << " respectively" << endl;

    return 0;

}

**OUTPUT:**



// 3. WAP to find simple and compound interest using simple inheritance?

#include <iostream>

#include <math.h>

using namespace std;

class A

{

public:

    int p, t;

    float r;

    A(int a, int b, float c)

    {

        p = a, t = b, r = c;

    }

    float SI()

    {

        float si = (p \* t \* r) / 100.0;

        cout << "Simple interest:" << si << endl;

    }

};

class B : public A

{

public:

    int n;

    B(int a, int b, float c, int d) : A(a, b, c)

    {

        n = d;

    }

    float CI()

    {

        float a = p \* (pow(1 + (r / n), (n \* t)));

        float ci = a - p;

        cout << "compound interest:" << ci << endl;

    }

};

int main()

{

    int p, t;

    float r;

    cout << "Enter principal amout,time and rate of interest:" << endl;

    cin >> p >> t >> r;

    int n;

    cout << "Enter time period for compound interest:" << endl;

    cin >> n;

    B b(p, t, r, n);

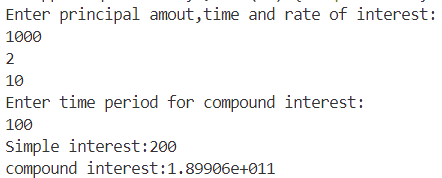
    b.SI();

    b.CI();

    return 0;

}

**OUTPUT:**



// 4. WAP to create an application for Student Assessment using inheritance?

// a) There are 5 Students in the class.

// b) Each student has attributes like Name , Roll No for unique ID and 5 subjects like C, C++, Java,

// Python, IoT. Full mark of each subject is 100 that is combination of 40 Marks for internal and 60 for

// external. CreditPoint of each subject is 3.

// c) For each subject, accept number of classes attendered by students and total number of class taken

// by teacher and find the ratio between these two estimate the internal mark of the subject.

// d) external mark will be received from the teacher.

// e) Find grade of each subject.

// f) Find SGPA of student.

// g) Display the result as per the IGIT result format

#include <iostream>

#include <string>

using namespace std;

// Base class for Student

class Student {

    protected:

        string name;

        int rollNo;

        int subjects[5];

        int internalMarks[5];

        int externalMarks[5];

        float creditPoints[5];

        float SGPA;

    public:

        Student(string n, int r) {

            name = n;

            rollNo = r;

        }

        virtual void getInternalMarks() = 0;

        virtual void getExternalMarks() = 0;

        virtual void calculateGrade() = 0;

        virtual void calculateSGPA() = 0;

        virtual void displayResult() = 0;

};

// Derived class for Student Assessment

class StudentAssessment : public Student {

    public:

        StudentAssessment(string n, int r) : Student(n, r) {}

        void getInternalMarks()  {

            cout << "Enter the number of classes attended and total classes for each subject:" << endl;

            for (int i = 0; i < 5; i++) {

                int attended, total;

                cout << "Subject " << i + 1 << ": ";

                cin >> attended >> total;

                internalMarks[i] = (attended \* 40) / total;

            }

        }

        void getExternalMarks() {

            cout << "Enter the external marks for each subject:" << endl;

            for (int i = 0; i < 5; i++) {

                cout << "Subject " << i + 1 << ": ";

                cin >> externalMarks[i];

            }

        }

        void calculateGrade() {

            for (int i = 0; i < 5; i++) {

                subjects[i] = internalMarks[i] + externalMarks[i];

                if (subjects[i] >= 90) {

                    creditPoints[i] = 4.0;

                } else if (subjects[i] >= 80) {

                    creditPoints[i] = 3.5;

                } else if (subjects[i] >= 70) {

                    creditPoints[i] = 3.0;

                } else if (subjects[i] >= 60) {

                    creditPoints[i] = 2.5;

                } else if (subjects[i] >= 50) {

                    creditPoints[i] = 2.0;

                } else {

                    creditPoints[i] = 0.0;

                }

            }

        }

        void calculateSGPA() {

            float totalCreditPoints = 0.0;

            float totalCredits = 0.0;

            for (int i = 0; i < 5; i++) {

                totalCreditPoints += creditPoints[i] \* 3;

                totalCredits += 3;

            }

            SGPA = totalCreditPoints / totalCredits;

        }

        void displayResult() {

            cout << "Student Name: " << name << endl;

            cout << "Registration Number: " << rollNo << endl;

            cout << "Grades in all subjects:" << endl;

            for (int i = 0; i < 5; i++) {

                cout << "[" << i + 1 << "] Subject " << i + 1 << ": " << subjects[i] << endl;

            }

            cout << "SGPA: " << SGPA << endl;

        }

};

int main() {

    const int numStudents = 5;

    cout << "Enter student name and registration number for student: " << endl;

    string name;

    int rollNo;

    cin >> name >> rollNo;

    StudentAssessment students(name, rollNo);

    students.getInternalMarks();

    students.getExternalMarks();

    students.calculateGrade();

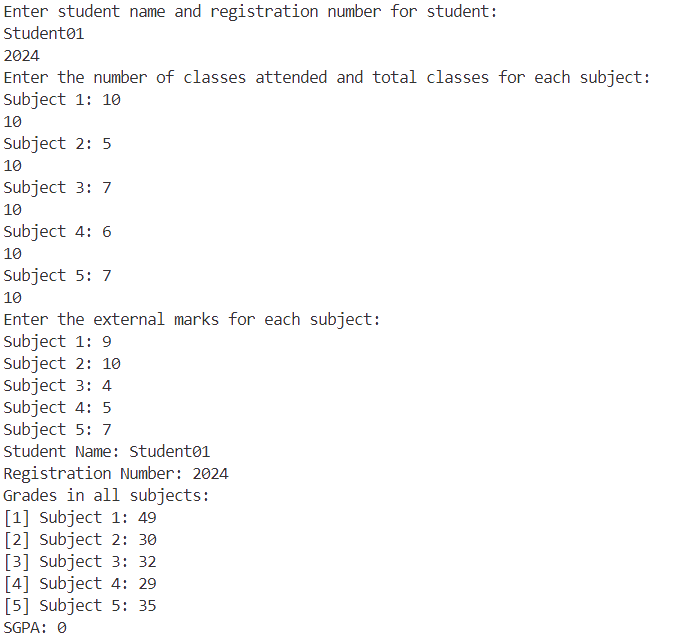
    students.calculateSGPA();

    students.displayResult();

    return 0;

}

**OUTPUT:**

****

// 5. WAP to create an banking application using inheritance and do following.

// a) Create current account of the user with user Id and minimum balance 500.

// b) Allow user to deposit, withdraw and check balance on the account.

// c) Allow user to open fixed deposit account with the user defined amount available on the current

// account and display the interest obtained quarterly and the final maturity value with the interest

// obtained.

// d) Allow user to open Rd deposit account with the user defined amount available on the current

// account and display the interest obtained quarterly and the final maturity value with the interest

// obtained. Allow user to deposit premium monthly manually or automatically from the account.

#include <iostream>

#include <string>

using namespace std;

// Base class for Bank Account

class BankAccount {

protected:

    string userId;

    double balance;

public:

    BankAccount(string id, double bal) {

        userId = id;

        balance = bal;

    }

    virtual void deposit(double amount) = 0;

    virtual void withdraw(double amount) = 0;

    virtual void checkBalance() = 0;

};

// Derived class for Current Account

class CurrentAccount : public BankAccount {

public:

    CurrentAccount(string id, double bal) : BankAccount(id, bal) {}

    void deposit(double amount) override {

        balance += amount;

        cout << "Deposit successful. New balance: " << balance << endl;

    }

    void withdraw(double amount) override {

        if (balance >= amount) {

            balance -= amount;

            cout << "Withdrawal successful. New balance: " << balance << endl;

        } else {

            cout << "Insufficient balance." << endl;

        }

    }

    void checkBalance() override {

        cout << "Current balance: " << balance << endl;

    }

};

// Derived class for Fixed Deposit Account

class FixedDepositAccount : public BankAccount {

private:

    double principalAmount;

    double interestRate;

    int tenure;

public:

    FixedDepositAccount(string id, double bal, double amount, double rate, int tenure) : BankAccount(id, bal) {

        principalAmount = amount;

        interestRate = rate;

        this->tenure = tenure;

    }

    void deposit(double amount) override {

        cout << "Fixed deposit account does not support deposit." << endl;

    }

    void withdraw(double amount) override {

        cout << "Fixed deposit account does not support withdrawal." << endl;

    }

    void checkBalance() override {

        double interest = principalAmount \* interestRate \* tenure / 100;

        double maturityValue = principalAmount + interest;

        cout << "Fixed deposit account details:" << endl;

        cout << "Principal amount: " << principalAmount << endl;

        cout << "Interest rate: " << interestRate << "%" << endl;

        cout << "Tenure: " << tenure << " months" << endl;

        cout << "Interest obtained: " << interest << endl;

        cout << "Maturity value: " << maturityValue << endl;

    }

};

// Derived class for Recurring Deposit Account

class RecurringDepositAccount : public BankAccount {

private:

    double principalAmount;

    double interestRate;

    int tenure;

public:

    RecurringDepositAccount(string id, double bal, double amount, double rate, int tenure) : BankAccount(id, bal) {

        principalAmount = amount;

        interestRate = rate;

        this->tenure = tenure;

    }

    void deposit(double amount) override {

        if (amount > 0) {

            balance += amount;

            cout << "Deposit successful. New balance: " << balance << endl;

        } else {

            cout << "Invalid deposit amount." << endl;

        }

    }

    void withdraw(double amount) override {

        cout << "Recurring deposit account does not support withdrawal." << endl;

    }

    void checkBalance() override {

        double interest = principalAmount \* interestRate \* tenure / 100;

        double maturityValue = principalAmount + interest;

        cout << "Recurring deposit account details:" << endl;

        cout << "Principal amount: " << principalAmount << endl;

        cout << "Interest rate: " << interestRate << "%" << endl;

        cout << "Tenure: " << tenure << " months" << endl;

        cout << "Interest obtained: " << interest << endl;

        cout << "Maturity value: " << maturityValue << endl;

    }

};

// Derived class for Loan

class Loan {

private:

    double amount;

    double interestRate;

    int tenure;

public:

    Loan(double amount, double rate, int tenure) {

        this->amount = amount;

        this->interestRate = rate;

        this->tenure = tenure;

    }

    void checkLoan() {

        double interest = amount \* interestRate \* tenure / 100;

        double totalAmount = amount + interest;

        cout << "Loan details:" << endl;

        cout << "Amount: " << amount << endl;

        cout << "Interest rate: " << interestRate << "%" << endl;

        cout << "Tenure: " << tenure << " months" << endl;

        cout << "Interest obtained: " << interest << endl;

        cout << "Total amount: " << totalAmount << endl;

    }

};

int main() {

    CurrentAccount currentAccount("1234567890", 500);

    FixedDepositAccount fixedDepositAccount("1234567891", 500, 10000, 5, 6);

    RecurringDepositAccount recurringDepositAccount("1234567892", 500, 10000, 5, 6);

    Loan loan(5000, 5, 6);

    cout << "1. Deposit" << endl;

        cout << "2. Withdraw" << endl;

        cout << "3. Check balance" << endl;

        cout << "4. Open fixed deposit account" << endl;

        cout << "5. Open recurring deposit account" << endl;

        cout << "6. Take loan" << endl;

        cout << "7. Exit" << endl;

    int choice;

    while (true) {

        cout << "Enter your choice: ";

        cin >> choice;

        switch (choice) {

            case 1:

                cout << "Enter the amount to deposit: ";

                double depositAmount;

                cin >> depositAmount;

                currentAccount.deposit(depositAmount);

                break;

            case 2:

                cout << "Enter the amount to withdraw: ";

                double withdrawAmount;

                cin >> withdrawAmount;

                currentAccount.withdraw(withdrawAmount);

                break;

            case 3:

                currentAccount.checkBalance();

                break;

            case 4:

                cout << "Enter the principal amount: ";

                double fixedDepositAmount;

                cin >> fixedDepositAmount;

                fixedDepositAccount.deposit(fixedDepositAmount);

                break;

            case 5:

                cout << "Enter the principal amount: ";

                double recurringDepositAmount;

                cin >> recurringDepositAmount;

                recurringDepositAccount.deposit(recurringDepositAmount);

                break;

            case 6:

                loan.checkLoan();

                break;

            case 7:

                return 0;

            default:

                cout << "Invalid choice. Please try again." << endl;

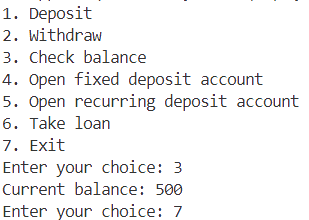
        }

    }

    return 0;

}

**OUTPUT:**

****

5 //  WAP to find area, perimeter and volume of a Rectangular, square and Circular Box using multilevel inheritance and function overriding?

#include <iostream>

#include <cmath>

using namespace std;

// Base class for geometric shapes

class Shape {

    public:

        virtual int calculateArea() = 0;

        virtual int calculatePerimeter() = 0;

        virtual int calculateVolume() = 0;

};

// Base class for rectangular and square shapes

class RectangularShape : public Shape {

    protected:

        int length;

        int width;

    public:

        RectangularShape(int l, int w) : length(l), width(w) {}

        int calculateArea() override {

            return length \* width;

        }

        int calculatePerimeter() override {

            return 2 \* (length + width);

        }

        int calculateVolume() override {

            return length \* width;

        }

};

// Base class for circular shapes

class CircularShape : public Shape {

    protected:

        int radius;

    public:

        CircularShape(int r) : radius(r) {}

        int calculateArea() override {

            return static\_cast<int>(M\_PI \* radius \* radius);

        }

        int calculatePerimeter() override {

            return static\_cast<int>(2 \* M\_PI \* radius);

        }

        int calculateVolume() override {

            return static\_cast<int>(M\_PI \* radius \* radius \* radius);

        }

};

// Derived class for square

class Square : public RectangularShape {

    public:

        Square(int s) : RectangularShape(s, s) {}

};

// Derived class for rectangle

class Rectangle : public RectangularShape {

    public:

        Rectangle(int l, int w) : RectangularShape(l, w) {}

};

// Derived class for circle

class Circle : public CircularShape {

    public:

        Circle(int r) : CircularShape(r) {}

};

int main() {

    // Create instances of shapes

    Square square(5);

    Rectangle rectangle(4, 6);

    Circle circle(3);

    // Calculate and print properties

    cout << "Square Area: " << square.calculateArea() << endl;

    cout << "Square Perimeter: " << square.calculatePerimeter() << endl;

    cout << "Square Volume: " << square.calculateVolume() << endl;

    cout << "Rectangle Area: " << rectangle.calculateArea() << endl;

    cout << "Rectangle Perimeter: " << rectangle.calculatePerimeter() << endl;

    cout << "Rectangle Volume: " << rectangle.calculateVolume() << endl;

    cout << "Circle Area: " << circle.calculateArea() << endl;

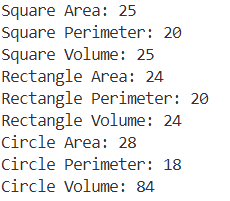
    cout << "Circle Perimeter: " << circle.calculatePerimeter() << endl;

    cout << "Circle Volume: " << circle.calculateVolume() << endl;

    return 0;

}

**OUTPUT:**

****

// 6. WAP to find simple interest provided by bank on the available amount on the saving account by

// various banks SBI, Union, HDFC, ICICI with rate of interest 6.5,6.5, 7.4, 7.9 respectively using

// function overriding and virtual class. Minimum rate of interest is 5.4 and mechanism to compute

// simple and compound interest is set by RBI.

#include <iostream>

#include <iomanip>

#include<cmath>

using namespace std;

// Base class for Bank

class Bank {

protected:

    double amount;

    double rateOfInterest;

public:

    Bank(double amt, double rate) {

        amount = amt;

        rateOfInterest = rate;

    }

    virtual double getSimpleInterest(int duration) = 0;

    virtual double getCompoundInterest(int duration) = 0;

};

// Derived class for SBI

class SBI : public Bank {

public:

    SBI(double amt) : Bank(amt, 6.5) {}

    double getSimpleInterest(int duration) override {

        return (amount \* rateOfInterest \* duration) / 100;

    }

    double getCompoundInterest(int duration) override {

        return amount \* pow(1 + rateOfInterest / 100, duration) - amount;

    }

};

// Derived class for Union Bank

class UnionBank : public Bank {

public:

    UnionBank(double amt) : Bank(amt, 6.5) {}

    double getSimpleInterest(int duration) override {

        return (amount \* rateOfInterest \* duration) / 100;

    }

    double getCompoundInterest(int duration) override {

        return amount \* pow(1 + rateOfInterest / 100, duration) - amount;

    }

};

// Derived class for HDFC Bank

class HDFC : public Bank {

public:

    HDFC(double amt) : Bank(amt, 7.4) {}

    double getSimpleInterest(int duration) override {

        return (amount \* rateOfInterest \* duration) / 100;

    }

    double getCompoundInterest(int duration) override {

        return amount \* pow(1 + rateOfInterest / 100, duration) - amount;

    }

};

// Derived class for ICICI Bank

class ICICI : public Bank {

public:

    ICICI(double amt) : Bank(amt, 7.9) {}

    double getSimpleInterest(int duration) override {

        return (amount \* rateOfInterest \* duration) / 100;

    }

    double getCompoundInterest(int duration) override {

        return amount \* pow(1 + rateOfInterest / 100, duration) - amount;

    }

};

int main() {

    double amount;

    int duration;

    cout << "Enter the available amount in the savings account: ";

    cin >> amount;

    cout << "Enter the duration (in years): ";

    cin >> duration;

    Bank\* banks[] = {

        new SBI(amount),

        new UnionBank(amount),

        new HDFC(amount),

        new ICICI(amount)

    };

    cout << "Bank\tSimple Interest\tCompound Interest" << endl;

    cout << "----\t---------------\t-----------------" << endl;

    for (int i = 0; i < 4; i++) {

        cout << (i == 0 ? "SBI" : (i == 1 ? "Union" : (i == 2 ? "HDFC" : "ICICI"))) << "\t"

             << fixed << setprecision(2) << banks[i]->getSimpleInterest(duration) << "\t\t"

             << banks[i]->getCompoundInterest(duration) << endl;

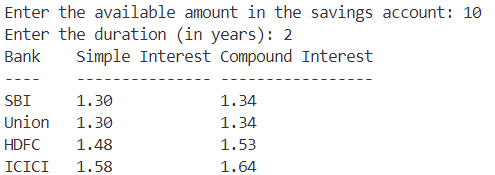
        delete banks[i];

    }

    return 0;

}

**OUTPUT:**



// WAP to display who is the owner of bike in which parent and child has a bike of same brand using

// using function overriding and virtual class?

#include <iostream>

#include <string>

using namespace std;

// Base class for Bike

class Bike

{

protected:

    string brand;

    string owner;

public:

    Bike(string b, string o)

    {

        brand = b;

        owner = o;

    }

    virtual void displayOwner()

    {

        cout << "Owner of " << brand << " bike: " << owner << endl;

    }

};

// Derived class for Parent

class Parent : public Bike

{

public:

    Parent(string b, string o) : Bike(b, o) {}

    void displayOwner() override

    {

        cout << "Parent owns " << brand << " bike." << endl;

    }

};

// Derived class for Child

class Child : public Bike

{

public:

    Child(string b, string o) : Bike(b, o) {}

    void displayOwner() override

    {

        cout << "Child owns " << brand << " bike." << endl;

    }

};

int main()

{

    Parent parent("Honda", "Minakshi");

    Child child("Honda", "Dibya");

    parent.displayOwner();

    child.displayOwner();

    return 0;

}

**OUTPUT:**



**Friend class and friend function**

// 1. WAP to create a simple calculator for operations such as +, -, \*, /,%, factorial of largest among two

// numbers using friend class?

#include <iostream>

#include <cmath>

using namespace std;

class Calculator

{

private:

    int num1, num2;

public:

    void setNumbers(int a, int b)

    {

        num1 = a;

        num2 = b;

    }

    friend class Operations;

};

class Operations

{

public:

    static int add(Calculator &calc)

    {

        return calc.num1 + calc.num2;

    }

    static int subtract(Calculator &calc)

    {

        return calc.num1 - calc.num2;

    }

    static int multiply(Calculator &calc)

    {

        return calc.num1 \* calc.num2;

    }

    static int divide(Calculator &calc)

    {

        if (calc.num2 == 0)

        {

            cout << "Error: Division by zero" << endl;

            return 0;

        }

        return calc.num1 / calc.num2;

    }

    static int modulo(Calculator &calc)

    {

        if (calc.num2 == 0)

        {

            cout << "Error: Division by zero" << endl;

            return 0;

        }

        return calc.num1 % calc.num2;

    }

    static int factorial(Calculator &calc)

    {

        int larger = (calc.num1 > calc.num2) ? calc.num1 : calc.num2;

        if (larger < 0)

        {

            cout << "Error: Factorial of a negative number is not defined" << endl;

            return 0;

        }

        int fact = 1;

        for (int i = 1; i <= larger; i++)

        {

            fact \*= i;

        }

        return fact;

    }

};

int main()

{

    Calculator calc;

    int num1, num2;

    cout << "Enter two numbers: ";

    cin >> num1 >> num2;

    calc.setNumbers(num1, num2);

    cout << "Addition: " << Operations::add(calc) << endl;

    cout << "Subtraction: " << Operations::subtract(calc) << endl;

    cout << "Multiplication: " << Operations::multiply(calc) << endl;

    cout << "Division: " << Operations::divide(calc) << endl;

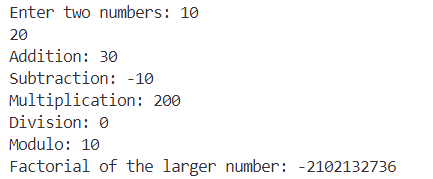
    cout << "Modulo: " << Operations::modulo(calc) << endl;

    cout << "Factorial of the larger number: " << Operations::factorial(calc) << endl;

    return 0;

}

**OUTPUT:**

****

// 2. WAP to find area and perimeter of rectangle and square using function overloading and friend class?

#include <iostream>

using namespace std;

class Shape

{

private:

    int length, width;

public:

    void setDimensions(int l, int w)

    {

        length = l;

        width = w;

    }

    friend int area(Shape);

    friend int perimeter(Shape);

};

int area(Shape s)

{

    return s.length \* s.width;

}

int perimeter(Shape s)

{

    return 2 \* (s.length + s.width);

}

int main()

{

    Shape rectangle, square;

    int choice, l, w;

    cout << "1. Rectangle" << endl;

    cout << "2. Square" << endl;

    cout << "3. Exit" << endl;

    while (true)

    {

        cout << "Enter your choice: ";

        cin >> choice;

        switch (choice)

        {

        case 1:

            cout << "Enter length and width of rectangle: ";

            cin >> l >> w;

            rectangle.setDimensions(l, w);

            cout << "Area of rectangle: " << area(rectangle) << endl;

            cout << "Perimeter of rectangle: " << perimeter(rectangle) << endl;

            break;

        case 2:

            cout << "Enter side of square: ";

            cin >> l;

            square.setDimensions(l, l);

            cout << "Area of square: " << area(square) << endl;

            cout << "Perimeter of square: " << perimeter(square) << endl;

            break;

        case 3:

            return 0;

        default:

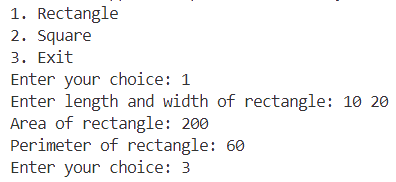
            cout << "Invalid choice. Please try again." << endl;

        }

    }

}

**OUTPUT:**

****

// 3. WAP to difference between sum of all even and odd numbers in an array of 10 numbers using friend

// function?

#include <iostream>

using namespace std;

class ArraySum {

private:

    int arr[10];

public:

    void getArray() {

        cout << "Enter 10 numbers: ";

        for (int i = 0; i < 10; i++) {

            cin >> arr[i];

        }

    }

    friend int findDifference(ArraySum);

};

int findDifference(ArraySum obj) {

    int evenSum = 0, oddSum = 0;

    for (int i = 0; i < 10; i++) {

        if (obj.arr[i] % 2 == 0) {

            evenSum += obj.arr[i];

        } else {

            oddSum += obj.arr[i];

        }

    }

    return evenSum - oddSum;

}

int main() {

    ArraySum obj;

    obj.getArray();

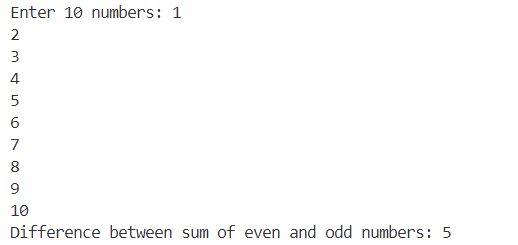
    int diff = findDifference(obj);

    cout << "Difference between sum of even and odd numbers: " << diff << endl;

    return 0;

}

**OUTPUT:**

****

// 4. Write a program to swap two numbers without using 3rd variable and with using 3rd variable using

// friend function?

#include <iostream>

using namespace std;

class Swap

{

private:

    int a, b;

public:

    void setValues(int x, int y)

    {

        a = x;

        b = y;

        cout << "Before swapping: num1 = " << a << ", num2 = " << b << endl;

    }

    friend void swapNumbers(Swap &);

};

void swapNumbers(Swap &obj)

{

    obj.a = obj.a + obj.b;

    obj.b = obj.a - obj.b;

    obj.a = obj.a - obj.b;

    cout << "After swapping: num1 = " << obj.a << ", num2 = " << obj.b << endl;

}

int main()

{

    Swap obj;

    int num1, num2;

    cout << "Enter two numbers: ";

    cin >> num1 >> num2;

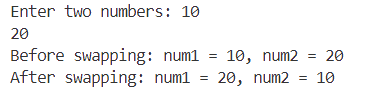
    obj.setValues(num1, num2);

    swapNumbers(obj);

    return 0;

}

**OUTPUT:**

****

#include <iostream>

using namespace std;

class Box

{

private:

    int ln, br, ht;

public:

    Box() : ln(0), br(0), ht(0) {} // Default constructor

    Box(int a, int b, int c) : ln(a), br(b), ht(c) {}

    Box operator+(const Box &o)

    {

        Box t;

        t.ln = ln + o.ln;

        t.br = br + o.br;

        t.ht = ht + o.ht;

        return t;

    }

    friend void volume(const Box &b);

};

void volume(const Box &b)

{

    int v = b.ln \* b.br \* b.ht;

    cout << "Volume of the box is: " << v << endl;

}

int main()

{

    cout << "Box1..................." << endl;

    int a, b, c;

    cout << "Enter length, breadth and height of the box1:" << endl;

    cin >> a >> b >> c;

    Box b1(a, b, c);

    volume(b1);

    cout << "Box2..................." << endl;

    int d, e, f;

    cout << "Enter length, breadth and height of the box2:" << endl;

    cin >> d >> e >> f;

    Box b2(d, e, f);

    volume(b2);

    cout << "Box3..................." << endl;

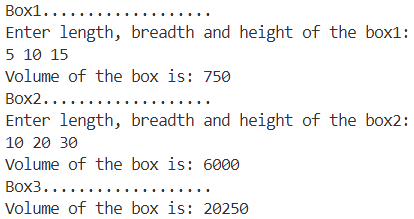
    Box b3 = b1 + b2;

    volume(b3);

    return 0;

}

**OUTPUT:**

****