

Winter Winning Camp Assignment(1)

NAME : - Harshit Bansal
UID : - 22BCS15323

SUBJECT: C++ Programming

DAY- 1

SOLUTION 1:

Prime number check:

```
#include <iostream>
using namespace std;
int main()
{
    int n, i, m=0, flag=0;
    cout << "Enter the Number to check Prime: ";
    cin >> n;
    m=n/2;
    for(i = 2; i <= m; i++)
    {
        if(n % i == 0)
        {
            cout<<"Number is not Prime."<<endl;
            flag=1;
            break;
        }
    }
    if (flag==0)
        cout << "Number is Prime."<<endl;
    return 0;
}
```

OUTPUT:

```
pp -o tempCodeRunnerFile } ; if ($?) { .\tempCodeRunnerFile }
Enter the Number to check Prime: 7
Number is Prime.
```

SOLUTION 2:

Odd Number Sum:

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

int sumoddnum(int n) {
    int i, sum = 0;

    for (i = 1; i <= n; i++) {
        if (i % 2 == 1) {
            cout << i << ", ";
            sum = sum + i;
        }
    }
}
```

Winter Winning Camp Assignment(1)

```

        return sum;
    }

int main() {
    int n;
    cin >> n;
    int sum = sumodddnum(n);
    cout << endl << sum;
    return 0;
}

```

OUTPUT:

```

oddupn } ; if ($?) { .\sumoddupn }
10
1, 3, 5, 7, 9,
25

```

SOLUTION 3:

Print Multiplication Table of a Number:

```

#include<iostream>

using namespace std;

int main(){

    int n;

    cout<<"enter the number that the table you want to print "<<endl;
    cin>>n;

    for (int i=1 ; i<=10 ; i++){
        int table = n*i;
        cout<<n<<"* "<<i<<"="<<table<<endl;

    }

    return 0;

}

```

OUTPUT:

```

enter the number that the table you want to print
3
3*1=3
3*2=6
3*3=9
3*4=12
3*5=15
3*6=18
3*7=21
3*8=24
3*9=27
3*10=30

```

Winter Winning Camp Assignment(1)

SOLUTION 4:

Reverse a number:

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

int reverseNumber(int number) {
    string numberStr = to_string(number);

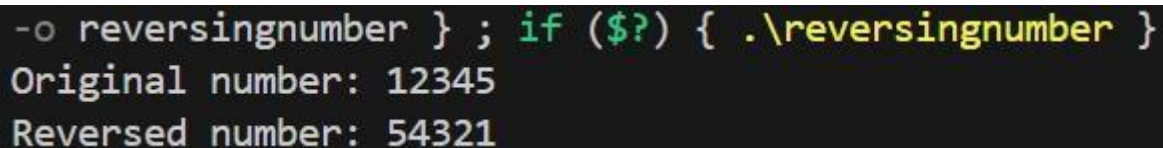
    reverse(numberStr.begin(), numberStr.end());

    int reversedNumber = stoi(numberStr);

    return reversedNumber;
}

int main() {
    int originalNumber = 12345;
    int reversedNumber = reverseNumber(originalNumber);
    cout << "Original number: " << originalNumber << endl;
    cout << "Reversed number: " << reversedNumber << endl;
    return 0;
}
```

OUTPUT:



```
-o reversingnumber } ; if ($?) { .\reversingnumber }
Original number: 12345
Reversed number: 54321
```

SOLUTION 5:

Palindrome number:

```
#include<iostream>
using namespace std;
int palidromecheck(int n){
    int reverse =0;
    int temp= n;
    while(temp!=0){
        reverse=(reverse*10 )+(temp%10);
        temp= temp/10;
    }
    return (reverse == n);
}

int main(){
    int n;
    cin >>n;
    if (palidromecheck(n)==1){
        cout<<"the number is palindrome number"<<endl;
    }
    else{
```

Winter Winning Camp Assignment(1)

```
        cout<<"you are wrong the number is not a palindrome"<<endl;
    }
    return 0;
}
```

OUTPUT:

```
pp -o tempCodeRunnerFile } ; if ($?) { .\tempCodeRunnerFile }
12321
the number is palindrome number
```

SOLUTION 6:

Area check:

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

void area(int l , int b){
    int rectangle_area = l*b;

    cout<<"Area of rectangle"<<endl<<rectangle_area<<endl;
}
void area(int r){
    float circle_area = 3.141*r*r;
    cout<<"area of circle"<<endl<<circle_area<<endl;
}
void area(double x, double h){
    double triangle = 0.5*h*x;
    cout<<"area of triangle"<<endl<<triangle;
}

int main(){
    area(2,2);
    area(3);
    area(5.3,3.2);
}
```

OUTPUT:

```
reacompares } ; if ($?) { .\areacompares }
Area of rectangle
4
area of circle
28.269
area of triangle
8.48
```

SOLUTION 7:

Hierarchical Overloading

```
#include <iostream>
using namespace std;
int calculateSalary(int stipend) {
    return stipend;
}
```

Winter Winning Camp Assignment(1)

```
int calculateSalary(int baseSalary, int bonuses) {
    return baseSalary + bonuses;
}
int calculateSalary(int baseSalary, int bonuses, int incentives)
{
    return baseSalary + bonuses + incentives;
}

int main() {
    int stipend, baseSalary, bonuses, incentives;
    cout << "Enter stipend for Intern: ";
    cin >> stipend;
    cout << "Intern Salary: " << calculateSalary(stipend) <<
endl;
    cout << "Enter base salary and bonuses for Regular Employee:
";
    cin >> baseSalary >> bonuses;
    cout << "Employee Salary: " << calculateSalary(baseSalary,
bonuses) << endl;

    cout << "Enter base salary, bonuses, and incentives for
Manager: ";
    cin >> baseSalary >> bonuses >> incentives;
    cout << "Manager Salary: " << calculateSalary(baseSalary,
bonuses, incentives) << endl;

    return 0;
}
```

```
hy } ; if ($?) { .\hierchy }
Enter stipend for Intern: 2000
Intern Salary: 2000
Enter base salary and bonuses for Regular Employee: 200000
5000
Employee Salary: 205000
Enter base salary, bonuses, and incentives for Manager: 300000
6000
2500
Manager Salary: 308500
```

SOLUTION 8:

Polymorphism of shape area

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

class Shape {
public:
    virtual double area() const = 0;
    virtual ~Shape() {}
};
```

Winter Winning Camp Assignment(1)

```
class Circle : public Shape {
private:
    double radius;
public:
    Circle(double r) : radius(r) {}
    double area() const override {
        return 3.14159 * radius * radius;
    }
};

class Rectangle : public Shape {
private:
    double length, breadth;
public:
    Rectangle(double l, double b) : length(l), breadth(b) {}
    double area() const override {
        return length * breadth;
    }
};

class Triangle : public Shape {
private:
    double base, height;
public:
    Triangle(double b, double h) : base(b), height(h) {}
    double area() const override {
        return 0.5 * base * height;
    }
};

int main() {
    double radius, length, breadth, base, height;
    cout << "Enter radius for Circle: ";
    cin >> radius;
    Shape *circle = new Circle(radius);
    cout << "Circle Area: " << circle->area() << endl;

    cout << "Enter length and breadth for Rectangle: ";
    cin >> length >> breadth;
    Shape *rectangle = new Rectangle(length, breadth);
    cout << "Rectangle Area: " << rectangle->area() << endl;
    cout << "Enter base and height for Triangle: ";
    cin >> base >> height;
    Shape *triangle = new Triangle(base, height);
    cout << "Triangle Area: " << triangle->area() << endl;
    delete circle;
    delete rectangle;
    delete triangle;
    return 0;
}
```

OUTPUT:

```
pp -o tempCodeRunnerFile } ; if ($?) { .\tempCodeRunnerFile }
Enter radius for Circle: 4.6
Circle Area: 66.476
Enter length and breadth for Rectangle: 77
55
Rectangle Area: 4235
Enter base and height for Triangle: 32
4
Triangle Area: 64
```

Winter Winning Camp Assignment(1)

SOLUTION 9:

Multi-level stimulation of vehical class

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

class Vehicle {
protected:
    string brand;
    string model;
    double mileage;

public:
    Vehicle(string b, string m, double mil) : brand(b), model(m),
    mileage(mil) {}

    virtual void display() const {
        cout << "Vehicle: " << brand << " " << model << endl;
        cout << "Mileage: " << mileage << endl;
    }
};

class Car : public Vehicle {
protected:
    double fuel;
    double distance;

public:
    Car(string b, string m, double mil, double f, double d) : Vehicle(b, m,
    mil), fuel(f), distance(d) {}

    double calculateFuelEfficiency() const {
        return distance / fuel;
    }

    void display() const override {
        Vehicle::display();
        cout << "Fuel Efficiency: " << calculateFuelEfficiency() << "
    miles/gallon" << endl;
    }
};

class ElectricCar : public Car {
private:
    double batteryCapacity;
    double efficiency;

public:
    ElectricCar(string b, string m, double mil, double bc, double eff)
        : Car(b, m, mil, 0, 0), batteryCapacity(bc), efficiency(eff) {}

    double calculateRange() const {
        return batteryCapacity * efficiency;
    }

    void display() const override {
        Vehicle::display();
        cout << "Range: " << calculateRange() << " miles" << endl;
    }
};
```

Winter Winning Camp Assignment(1)

```
int main() {
    int vehicleType;
    string brand, model;
    double mileage;

    cout << "Enter Vehicle Type (1 for Car, 2 for Electric Car): ";
    cin >> vehicleType;
    cout << "Enter Brand: ";
    cin >> brand;
    cout << "Enter Model: ";
    cin >> model;
    cout << "Enter Mileage: ";
    cin >> mileage;

    if (vehicleType == 1) {
        double fuel, distance;
        cout << "Enter Fuel (in gallons): ";
        cin >> fuel;
        cout << "Enter Distance Covered (in miles): ";
        cin >> distance;

        Car car(brand, model, mileage, fuel, distance);
        car.display();
    } else if (vehicleType == 2) {
        double batteryCapacity, efficiency;
        cout << "Enter Battery Capacity (in kWh): ";
        cin >> batteryCapacity;
        cout << "Enter Efficiency (in miles per kWh): ";
        cin >> efficiency;

        ElectricCar electricCar(brand, model, mileage, batteryCapacity,
efficiency);
        electricCar.display();
    } else {
        cout << "Invalid vehicle type entered." << endl;
    }

    return 0;}
```

```
al } ; if ($?) { .\vehical }
Enter Vehicle Type (1 for Car, 2 for Electric Car): 1
Enter Brand: Toyota
Enter Model: Corolla
Enter Mileage: 30000
Enter Fuel (in gallons): 15
Enter Distance Covered (in miles): 300
Vehicle: Toyota Corolla
Mileage: 30000
Fuel Efficiency: 20 miles/gallon
```