

Full name: Nguyen Xuan Chu

Student code: 240053

## Table of Contents

Solution .....	3
Question 1:.....	3
Result of question 1: .....	4
Question 2:.....	4
Result of question 2: .....	6
Question 3:.....	6
Result of question 3: .....	8
Question 4:.....	8
Result of question 4: .....	10
Question 5:.....	10
Result of question 5: .....	12
Question 6:.....	12
Result of question 6: .....	14
Question 7:.....	14
Result of question 7: .....	15
Question 8:.....	15
Result of question 8: .....	17
Question 9:.....	17
Result of question 9: .....	19
Question 10:.....	19
Result of question 10: .....	22
Question 11:.....	22
Result of question 11: .....	24
Question 12:.....	24
Result of question 12: .....	27
Question 13:.....	27
Result of question 13: .....	30
Question 14:.....	30
Result of question 14: .....	34
Question 15:.....	34

Result of question 15: .....	35
Question 16:.....	35
Result of question 16: .....	37
Question 17:.....	37
Result of question 17: .....	39
Question 18:.....	39
Result of question 18: .....	40
Question 19:.....	40
Result of question 19: .....	42
Question 20:.....	42
Result of question 20: .....	43
Question 21:.....	43
Result of question 21: .....	45
Question 22:.....	45
Result of question 22: .....	46
Question 23:.....	46
Result of question 23: .....	47
Question 24:.....	47
Result of question 24: .....	49
Question 25:.....	49
Result of question 25: .....	50
Question 26:.....	50
Result of question 26: .....	53
Question 27:.....	53
Result of question 27: .....	55
Question 28:.....	56
Result of question 28: .....	58
Question 29:.....	59
Result of question 29: .....	60
Question 30:.....	60
Result of question 30: .....	65

## Solution

### Question 1:

/\*

Question 1:

Write a program to check whether the given number is palindrome or not.

Make use of pure C++ standard I/O functions and header.

\*/

```
#include <iostream>
```

```
using namespace std;
```

```
string isPalindrome(int num)
```

```
{
```

```
    int rev = 0;
```

```
    int tmp = num;
```

```
    while(tmp != 0)
```

```
    {
```

```
        rev = rev*10 + tmp%10;
```

```
        tmp /= 10;
```

```
    }
```

```
    if (rev == num)
```

```
    {
```

```
        return "the given number is palindrome";
```

```
    }
```

```
    else
```

```
    {
```

```
        return "the given number is not palindrome";
```

```
    }
```

```
}
```

```
int main()
```

```

{
    int number;

    cout<<"Enter a number: ";
    cin>>number;
    cout<<isPalindrome(number)<<endl;

    return 0;
}

```

Result of question 1:

```

nxc@nxc-virtual-machine:~/Documents/cpp$ g++ q1.cpp
nxc@nxc-virtual-machine:~/Documents/cpp$ ./a.out
Enter a number: 32123
the given number is palindrome
nxc@nxc-virtual-machine:~/Documents/cpp$ ./a.out
Enter a number: 32101
the given number is not palindrome
nxc@nxc-virtual-machine:~/Documents/cpp$ S

```

Question 2:

```
/*
```

Question 2:

Write a C++ program to swap the values of 2 numbers without using the third variable, use class and take input from the user.

```
*/
```

```
#include<iostream>
```

```
using namespace std;
```

```
class Number
```

```
{
```

```
    int a, b;
```

```
    public:
```

```
    void getNumber()
```

```

{
    cout<<"Enter 1st number: ";
    cin>>a;
    cout<<"Enter 2nd number: ";
    cin>>b;
}

void dispNumber()
{
    cout<<"1st number: "<<a<<endl;
    cout<<"2nd number: "<<b<<endl;
}

void swapNumber()
{
    //a = 20
    //b = 10
    a = a + b; //a = 30
    b = a - b; //b = 30 - 10 = 20 and a = 30
    a = a - b;
}
};

int main()
{
    Number number;
    number.getNumber();
    cout<<"### before swapping 2 numbers"<<endl;
    number.dispNumber();
    cout<<"### after swapping 2 numbers"<<endl;
    number.swapNumber();
    number.dispNumber();
}

```

```
    return 0;
}
```

Result of question 2:

```
nxc@nxc-virtual-machine:~/Documents/cpp$ g++ q2.cpp
nxc@nxc-virtual-machine:~/Documents/cpp$ ./a.out
Enter 1st number: 20
Enter 2nd number: 10
### before swapping 2 numbers
1st number: 20
2nd number: 10
### after swapping 2 numbers
1st number: 10
2nd number: 20
nxc@nxc-virtual-machine:~/Documents/cpp$
```

Question 3:

```
/*
```

Question 3:

Write a C++ program to find the average score of the student, use class and get the inputs from the users

```
*/
```

```
#include<iostream>
```

```
using namespace std;
```

```
class Student
```

```
{
```

```
    private:
```

```
    int score[30];
```

```
    int num;
```

```
    float avg = 0;
```

```
    public:
```

```
    void getScore()
```

```
    {
```

```
        cout<<"The quatity of student's scores: ";
```

```
        cin>>num;
```

```

        cout<<"Enter "<<num<<" recent scores"<<endl;
        for (int i = 0; i < num; i++)
        {
            cin>>score[i];
        }
    }
    void calcAverage()
    {
        for (int i = 0; i < num; i++)
        {
            avg += score[i];
        }
        avg/=num;
        cout<<"The average score = "<<avg<<endl;
    }
    void dispScore()
    {
        cout<<"All recent scores"<<endl;
        for (int i = 0; i < num; i++)
        {
            cout<<score[i]<<" ";
        }
        cout<<endl;
    }
};

```

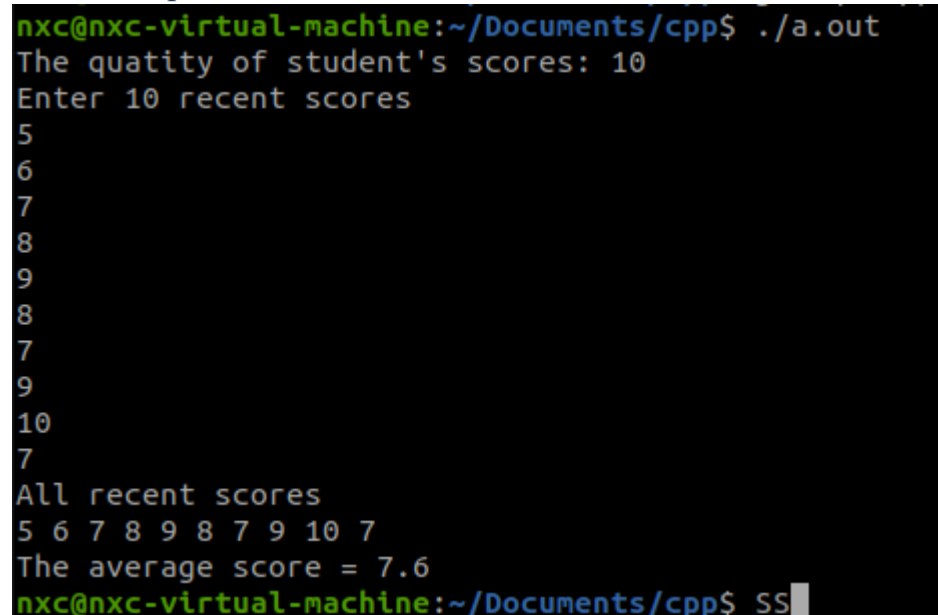
```

int main()
{
    Student student;
    student.getScore();
}

```

```
    student.dispScore();  
    student.calcAverage();  
    return 0;  
}
```

Result of question 3:



```
nxc@nxc-virtual-machine:~/Documents/cpp$ ./a.out  
The quaaty of student's scores: 10  
Enter 10 recent scores  
5  
6  
7  
8  
9  
8  
7  
9  
10  
7  
All recent scores  
5 6 7 8 9 8 7 9 10 7  
The average score = 7.6  
nxc@nxc-virtual-machine:~/Documents/cpp$ SS
```

Question 4:

```
/*
```

Question 4:

Write a C++ class called "Rectangle" that has private member variables for length and width. Include member functions to set the length and width, calculate the area and perimeter, and display the details of the rectangle.

```
*/
```

```
#include<iostream>
```

```
using namespace std;
```

```
class Rectangle
```

```
{
```

```
    private:
```

```
    int length, width;
```



```

public:
//constructor
Rectangle(int a, int b)
{
    length = a;
    width = b;
}
int calcArea()
{
    return length*width;
}
int calcPerimeter()
{
    return 2*(length+width);
}
void detailRectangle()
{
    cout<<"=== Detail of Rectangle ==="<<endl;
    cout<<"length = "<<length<<endl;
    cout<<"width = "<<width<<endl;
    cout<<"Area = "<<calcArea()<<endl;
    cout<<"Perimeter = "<<calcPerimeter()<<endl;
}
};

int main()
{
    Rectangle rect(2,3);
    rect.detailRectangle();
    return 0;
}

```

```
}
```

Result of question 4:

```
nxc@nxc-virtual-machine:~/Documents/cpp$ g++ q4.cpp
nxc@nxc-virtual-machine:~/Documents/cpp$ ./a.out
=== Detail of Rectangle ===
length = 2
width = 3
Area = 6
Perimeter = 10
nxc@nxc-virtual-machine:~/Documents/cpp$
```

Question 5:

```
/*
```

Question 5:

Create a class called "Employee" to represent an employee's information.

The class should have private member variables for:

- Employee ID (integer)
- Employee name (string)
- Salary (double)

The class should provide the following public member functions:

Include

- void getEmployeeInfo() a member function to get the employee details from the user
- void calculateYearlySalary(): Calculates and stores the yearly salary of the employee (assuming 12 months in a year).
- void displayEmployeeDetails(): Displays the employee's ID, name, monthly salary, and yearly salary.

Write a program that demonstrates the usage of the Employee class.

Create objects of the class, input the employee details and monthly salary, calculate the yearly salary, and display the employee details.

```
*/
```

```
#include<iostream>
```

```
using namespace std;
```

```
class Employee
{
    private:
        int empld;
        string empName;
        double empSalary[2];
    public:
        void getEmployeeInfo()
        {
            cout<<"  Information "<<endl;
            cout<<"Enter ID: ";
            cin>>empld;
            cout<<"Enter Name: ";
            cin>>empName;
            cout<<"Enter Monthly Salary: ";
            cin>>empSalary[0];
        }
        void calculateYearlySalary()
        {
            empSalary[1] = empSalary[0]*12;
        }
        void displayEmployeeDetails()
        {
            cout<<"  All Information"<<endl;
            cout<<"ID number: "<<empld<<endl;
            cout<<"Name: "<<empName<<endl;
            cout<<"Monthly Salary: "<<empSalary[0]<<endl;
            cout<<"Yearly Salary: "<<empSalary[1]<<endl;
        }
}
```

```
};
```

```
int main()
```

```
{
```

```
    Employee emp;
```

```
    emp.getEmployeeInfo();
```

```
    emp.calculateYearlySalary();
```

```
    emp.displayEmployeeDetails();
```

```
    return 0;
```

```
}
```

Result of question 5:

```
nxc@nxc-virtual-machine:~/Documents/cpp$ g++ q5.cpp
nxc@nxc-virtual-machine:~/Documents/cpp$ ./a.out
Information
Enter ID: 1
Enter Name: nguyen
Enter Monthly Salary: 100
All Information
ID number: 1
Name: nguyen
Monthly Salary: 100
Yearly Salary: 1200
```

Question 6:

```
/*
```

Question 6:

Create a class called "Circle" to represent a circle shape.

The class should have a private member variable for the radius (double).

The class should provide the following public member functions:

A constructor that initializes the radius.

- double calculateArea(): Calculates and returns the area of the circle.
- double calculateCircumference(): Calculates and returns the circumference of the circle.
- void displayCircleDetails(): Displays the radius, area, and circumference of the circle.

Write a program that demonstrates the usage of the Circle class.

Create objects of the class, input the radius of the circles, calculate the area and circumference, and display the circle details.

```
*/
```

```
#include<iostream>
```

```
using namespace std;
```

```
class Circle
```

```
{
```

```
    private:
```

```
    double radius;
```

```
    public:
```

```
    //constructor
```

```
    Circle(double rad):radius(rad){};
```

```
    double calculateArea()
```

```
    {
```

```
        return 3.14*radius*radius;
```

```
    }
```

```
    double calculateCircumference()
```

```
    {
```

```
        return 2*3.14*radius;
```

```
    }
```

```
    void displayCircleDetails()
```

```
    {
```

```
        cout<<"  Circle Detail"<<endl;
```

```
        cout<<"Radius: "<<radius<<endl;
```

```
        cout<<"Area: "<<calculateArea()<<endl;
```

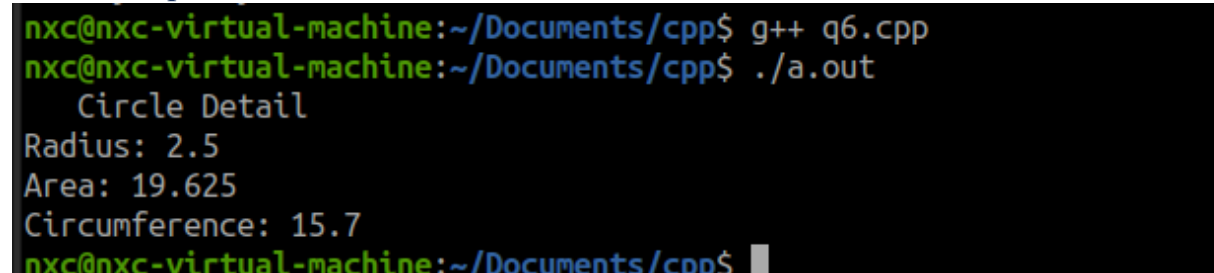
```
        cout<<"Circumference: "<<calculateCircumference()<<endl;
```

```
    }
```

```
};
```

```
int main()
{
    Circle c(2.5);
    c.displayCircleDetails();
}
```

Result of question 6:



```
nxc@nxc-virtual-machine:~/Documents/cpp$ g++ q6.cpp
nxc@nxc-virtual-machine:~/Documents/cpp$ ./a.out
    Circle Detail
Radius: 2.5
Area: 19.625
Circumference: 15.7
nxc@nxc-virtual-machine:~/Documents/cpp$
```

Question 7:

/\*

Question 7:

Create a class called "Student" to represent a student's information.

The class should have private member variables for student ID (integer) and name (string).

Implement a constructor that takes parameters to initialize the student ID and name.

Additionally, implement a destructor that prints a message indicating the student object is being removed.

\*/

```
#include<iostream>
```

```
using namespace std;
```

```
class Student
```

```
{
```

```
    private:
```

```
    int sID;
```

```
    string sName;
```

```
    public:
```

```

//constructor
Student(int id, string name)
{
    sID = id;
    sName = name;
}
~Student()
{
    cout<<"Student has ID: "<<sID<<, "
    <<"name: "<<sName<<" was deleted"<<endl;
}
};

int main()
{
    Student s(1, "Jack");
}

```

Result of question 7:

```

nxc@nxc-virtual-machine:~/Documents/cpp$ g++ q7.cpp
nxc@nxc-virtual-machine:~/Documents/cpp$ ./a.out
Student has ID: 1, name: Jack was deleted
nxc@nxc-virtual-machine:~/Documents/cpp$ 

```

Question 8:

/\*

Question 8:

Create a class of employee having name, employee ID and a performance index (pi) with years of experience(ye). The formula to calculate his pi is  $pi = ye * 10$ .

The name, ID and pi value should not be accessible outside the class.

Write a program using constructors of an object which will set the employee name, ID, pi value and print all the information in a legible and aesthetic manner.

Name, ID and ye is supplied at the time of object creation.

```
*/
```

```
#include<iostream>
```

```
using namespace std;
```

```
class Employee
```

```
{
```

```
    private:
```

```
    int eID;
```

```
    float eYE, ePI;
```

```
    string eName;
```

```
    public:
```

```
    //constructor
```

```
    Employee(int id, string name, float ye)
```

```
    {
```

```
        eID = id;
```

```
        eName = name;
```

```
        eYE = ye;
```

```
        ePI = 10*eYE;
```

```
        displayEmployeeDetails();
```

```
    }
```

```
    void displayEmployeeDetails()
```

```
    {
```

```
        cout<<" All Information"<<endl;
```

```
        cout<<"ID number: "<<eID<<endl;
```

```
        cout<<"Name: "<<eName<<endl;
```

```
        cout<<"Years of Experience: "<<eYE<<endl;
```

```
        cout<<"Performance Index: "<<ePI<<endl;
```

```
    }
```

```
};
```



```

int main()
{
    Employee emp(12, "John", 1.521);
    // emp.displayEmployeeDetails();
}

```

Result of question 8:

```

nxc@nxc-virtual-machine:~/Documents/cpp$ g++ q8.cpp
nxc@nxc-virtual-machine:~/Documents/cpp$ ./a.out
    All Information
ID number: 12
Name: John
Years of Experience: 1.521
Performance Index: 15.21
nxc@nxc-virtual-machine:~/Documents/cpp$

```

Question 9:

```
/*
```

Question 9:

Create a class of vehicles having the following properties, Wheels and Engine. Now a separate class of vehicles named Motorcycles will be created which have 2 wheels and 100 cc engine. A class of vehicles called cars will have 4 wheels and 1000cc engine.

The number of wheels and size of engine should be set at the time of object creation. Please create the base vehicle class and these two inheriting classes motorcycles and cars.

The inheriting classes should set all the values at the time of creation, also print them in a legible and aesthetic manner.

```
*/
```

```
#include<iostream>
```

```
using namespace std;
```

```
class Vehicles
```

```
{  
    public:  
    int wheels;  
    int engine;  
};
```

```
class Motorcycles: public Vehicles
```

```
{  
    public:  
    Motorcycles(int a, int b)  
    {  
        wheels = a;  
        engine = b;  
        displayInfomation();  
    }  
    void displayInfomation()  
    {  
        cout<<"Number of wheels: "<<wheels<<endl;  
        cout<<"Engine power: "<<engine<<"cc"<<endl;  
    }  
};
```

```
class Cars: public Vehicles
```

```
{  
    public:  
    Cars(int a, int b)  
    {
```

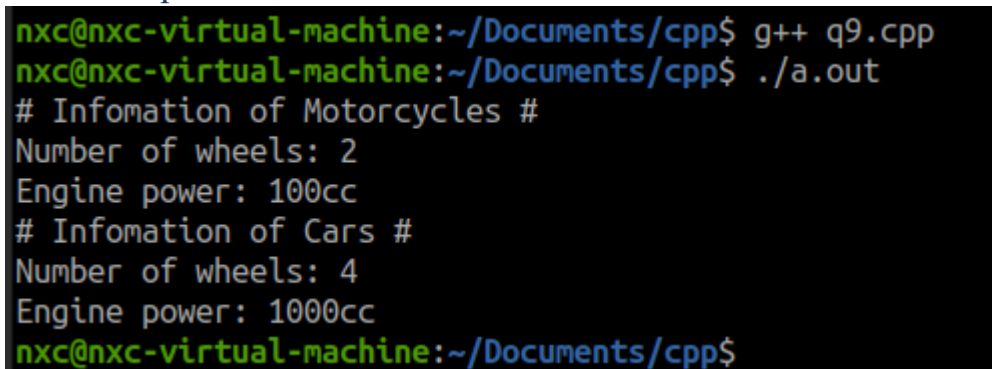
```

        wheels = a;
        engine = b;
        displayInfomation();
    }
    void displayInfomation()
    {
        cout<<"Number of wheels: "<<wheels<<endl;
        cout<<"Engine power: "<<engine<<"cc"<<endl;
    }
};

int main()
{
    cout<<"# Infomation of Motorcycles #"<<endl;
    Motorcycles m(2, 100);
    cout<<"# Infomation of Cars #"<<endl;
    Cars c(4, 1000);
}

```

Result of question 9:



```

nxc@nxc-virtual-machine:~/Documents/cpp$ g++ q9.cpp
nxc@nxc-virtual-machine:~/Documents/cpp$ ./a.out
# Infomation of Motorcycles #
Number of wheels: 2
Engine power: 100cc
# Infomation of Cars #
Number of wheels: 4
Engine power: 1000cc
nxc@nxc-virtual-machine:~/Documents/cpp$

```

Question 10:

/\*

Question 10:

Write a C++ program that reads two floating-point numbers from the user and performs division on them. Implement exception handling to catch any errors

that may occur during the division, such as invalid input or division by zero.

If an error occurs, display an appropriate error message to the user.

```
*/
```

```
#include<iostream>
```

```
using namespace std;
```

```
int main()
```

```
{
```

```
    float n1, n2;
```

```
    cout<<"Enter 1st floating-point number: ";
```

```
    try
```

```
    {
```

```
        cin>>n1;
```

```
        if (cin.fail())
```

```
        {
```

```
            throw 1;
```

```
        }
```

```
        //if no error with n1 then keep going to get n2
```

```
        cout<<"Enter 2nd floating-point number: ";
```

```
        try
```

```
        {
```

```
            cin>>n2;
```

```
            if (cin.fail())
```

```
            {
```

```
                throw 2;
```

```
            }
```

```
            if (n2 == 0)
```

```
            {
```

```
                throw 0;
```

```
    }  
    cout<<"division = "<<n1/n2<<endl;  
}  
catch (int b)  
{  
    if (b == 0)  
    {  
        cout<<"the 2nd number is 0"<<endl;  
    }  
    if (b == 2)  
    {  
        cout<<"invalid 2nd input"<<endl;  
    }  
}  
}  
catch (int a)  
{  
    cout<<"Invalid 1st input"<<endl;  
}  
  
return 0;  
}
```

Result of question 10:

```
nxc@nxc-virtual-machine:~/Documents/cpp$ g++ q10.cpp
nxc@nxc-virtual-machine:~/Documents/cpp$ ./a.out
Enter 1st floating-point number: a
Invalid 1st input
nxc@nxc-virtual-machine:~/Documents/cpp$ ./a.out
Enter 1st floating-point number: 12.5
Enter 2nd floating-point number: b
invalid 2nd input
nxc@nxc-virtual-machine:~/Documents/cpp$ ./a.out
Enter 1st floating-point number: 12.5
Enter 2nd floating-point number: 0
the 2nd number is 0
nxc@nxc-virtual-machine:~/Documents/cpp$ ./a.out
Enter 1st floating-point number: 12.5
Enter 2nd floating-point number: 2.0
division = 6.25
nxc@nxc-virtual-machine:~/Documents/cpp$
```

Question 11:

/\*

Question 11:

Write a C++ program that reads a sequence of integers from the user until a negative number is entered. Implement exception handling to catch any errors that may occur during the input process, such as non-integer values. If an error occurs, display an appropriate error message and discard the invalid input. Once the input is complete, calculate and display the sum of the entered integers.

\*/

```
#include<iostream>
```

```
using namespace std;
```

```
int main()
```

```
{
```

```
    int arr[100];
```

```
    int sum = 0;
```

```
    int i;
```

```

cout<<"Enter integer number until having negative number"<<endl;
for (i = 0; i < 100; i++)
{
    try
    {
        cin>>arr[i];
        if (cin.fail())
        {
            throw 0;
        }
        if (arr[i] < 0)
        {
            for (int j = 0; j <= i; j++)
            {
                sum += arr[j];
                // cout<<"sum = "<<sum<<endl;
            }
            cout<<"You have entered a negative number"<<endl;
            cout<<"Sum of all valid numbers = "<<sum<<endl;
            break;
        }
    }
}
catch(int e)
{
    cout<<"Invalid input "<<endl;
    cin.clear();
    cin.ignore(100, '\n');
    if (i > 0)
    {

```

```

        --i;
    }
    else
    {
        i = -1;
    }
}
}
return 0;
}

```

Result of question 11:

```

nxc@nxc-virtual-machine:~/Documents/cpp$ g++ q11.cpp
nxc@nxc-virtual-machine:~/Documents/cpp$ ./a.out
Enter integer number until having negative number
1
2
4
56
10
2
6
234
-234
You have entered a negative number
Sum of all valid numbers = 81
nxc@nxc-virtual-machine:~/Documents/cpp$

```

Question 12:

/\*

Question 12:

Write a C++ program which will capture user input of two strings, 1 and 2.

The program should capture the strings and throw exceptions on following conditions.

String 1 contains part of string 2, string 1 is of longer length.

String 2 contains part of string 1, string 2 is of longer length.



String 1 and string 2 are identical.

Every exception case should be captured and reported by exception handlers in a legible format.

Any other cases should not be captured or reported.

```
*/
```

```
#include<iostream>
```

```
#include<cstring>
```

```
using namespace std;
```

```
int main()
```

```
{
```

```
    char s1[100], s2[100];
```

```
    cout<<"Enter 1st string"<<endl;
```

```
    cin>>s1;
```

```
    cout<<"Enter 2nd string"<<endl;
```

```
    cin>>s2;
```

```
    try
```

```
    {
```

```
        if (strcmp(s1,s2) == 0)
```

```
        {
```

```
            throw 0;
```

```
        }
```

```
        if (strcmp(s1,s2) > 0)
```

```
        {
```

```
            throw 1;
```

```
        }
```

```
        if (strcmp(s1,s2) < 0)
```

```
        {
```

```
        throw 2;
    }
}
catch (int e)
{
    if (e == 0)
    {
        cout<<"String 1 and string 2 are identical"<<endl;
    }
    if (e == 1)
    {
        cout<<"String 1 contains part of string 2, string 1 is of longer length"<<endl;
    }
    if (e == 2)
    {
        cout<<"String 2 contains part of string 1, string 2 is of longer length"<<endl;
    }
}
return 0;
}
```

Result of question 12:

```
nxc@nxc-virtual-machine:~/Documents/cpp$ g++ q12.cpp
nxc@nxc-virtual-machine:~/Documents/cpp$ ./a.out
Enter 1st string
nguyenxuanchu
Enter 2nd string
nguyen
String 1 contains part of string 2, string 1 is of longer length
nxc@nxc-virtual-machine:~/Documents/cpp$ ./a.out
Enter 1st string
chu
Enter 2nd string
nguyenxuanchu
String 2 contains part of string 1, string 2 is of longer length
nxc@nxc-virtual-machine:~/Documents/cpp$ ./a.out
Enter 1st string
xuanchu
Enter 2nd string
xuanchu
String 1 and string 2 are identical
nxc@nxc-virtual-machine:~/Documents/cpp$
```

Question 13:

/\*

Question 13:

Devise a class to manage 4 students with name, roll no and performance index pi.

Pi is a private variable of the student class. Our routine should inspect each students pi and if any pi is below 50% then an exception should be thrown and the handler should report the students name and roll number along with the pi value.

\*/

```
#include<iostream>
```

```
using namespace std;
```

```
class Student
```

```
{
```

```
    public:
```

```
    string name;
```

```
    int roll_no;
```

```
int pi;
//constructor
Student(int r, string n, int p)
{
    roll_no = r;
    name = n;
    pi = p;
}
void disp()
{
    cout<<"Student Details"<<endl;
    cout<<"name: "<<name<<endl;
    cout<<"roll number: "<<roll_no<<endl;
    cout<<"performance index: "<<pi<<endl;
}
};
```

```
int main()
{
    Student s1(1, "Jack", 60);
    Student s2(2, "John", 40);
    Student s3(3, "James", 30);
    Student s4(4, "Janna", 70);
    Student *cptr;

    for (int i = 0; i < 4; i++)
    {
        switch(i)
        {
            case 0: cptr = &s1;
```

```

        break;
    case 1: cptr = &s2;
        break;
    case 2: cptr = &s3;
        break;
    case 3: cptr = &s4;
        break;
}
try
{
    if (cptr->pi < 50)
    {
        throw 0;
    }
}
catch (int e)
{
    cout<<"!!! Student has PI lower then 50%"<<endl;
    cptr->disp();
    cout<<endl;
}
}
return 0;
}

```

Result of question 13:

```
nxc@nxc-virtual-machine:~/Documents/cpp$ g++ q13.cpp
nxc@nxc-virtual-machine:~/Documents/cpp$ ./a.out
!!! Student has PI lower then 50%
Student Details
name: John
roll number: 2
performance index: 40

!!! Student has PI lower then 50%
Student Details
name: James
roll number: 3
performance index: 30
nxc@nxc-virtual-machine:~/Documents/cpp$
```

Question 14:

/\*

Question 14:

We have a class of students. 4 students are there. Our program is calculating the students exam scores. Our class should have a method which will get the score value from user for each student and throw an exception if the score is below 50%, throw an exception if any value over than 100 or less than zero is entered.

Exception should be thrown when any alphabet is entered. Our handler should report each exception case. And report the students name, roll and score for each case.

\*/

```
#include<iostream>
```

```
using namespace std;
```

```
class Student
```

```
{
```

```
    private:
```

```
    int roll_no;
```

```
    string name;
```

```
    int score;
```

```

public:
//constructor
Student(int rn, string n, int sc)
{
    roll_no = rn;
    name = n;
    score = sc;
}
void set_score()
{
    cout<<"enter "<<name<<"s score: ";
    cin>>score;
    // return cin.fail();
}
int get_score()
{
    return score;
}
void disp()
{
    cout<<"### Student Detail"<<endl;
    cout<<"roll number: "<<roll_no<<endl;
    cout<<"name: "<<name<<endl;
    cout<<"score: "<<score<<endl;
}
};

int main()
{
    bool f;

```

```
Student s1(1, "Jack", 50);
Student s2(2, "John", 50);
Student s3(1, "James", 50);
Student s4(1, "Janna", 50);
Student *cptr;
```

```
for (int i =0; i <4; i++)
{
    switch(i)
    {
        case 0: cptr = &s1;
            break;
        case 1: cptr = &s2;
            break;
        case 2: cptr = &s3;
            break;
        case 3: cptr = &s4;
            break;
    }
    try
    {
        cptr->set_score();
        if (cin.fail())
        {
            throw -1;
        }
        if (cptr->get_score()<0)
        {
            throw 0;
        }
    }
}
```



```

    if (cptr->get_score()<50)
    {
        throw 50;
    }
    if (cptr->get_score()>100)
    {
        throw 100;
    }
}
catch (int e)
{
    if (e == -1)
    {
        cin.clear();          //clear fail flag
        cin.ignore(1000,'\n'); //ignore 1000 characters in buffer
        cout<<"Invalid input"<<endl;
    }
    if (e == 0)
    {
        cout<<"score is less than 0%"<<endl;
    }
    if (e == 50)
    {
        cout<<"score is less than 50%"<<endl;
    }
    if (e == 100)
    {
        cout<<"score is greater than 100%"<<endl;
    }
    cptr->disp();
}

```

```

    }
}
return 0;
}

```

Result of question 14:

```

ual-machine:~/Documents/cpp$ g++ q14.cpp
nxc@nxc-virtual-machine:~/Documents/cpp$ ./a.out
enter Jack's score: dhasasd
Invalid input
### Student Detail
roll number: 1
name: Jack
score: 0
enter John's score: 30
score is less than 50%
### Student Detail
roll number: 2
name: John
score: 30
enter James's score: 120
score is greater than 100%
### Student Detail
roll number: 1
name: James
score: 120
enter Janna's score: 90
nxc@nxc-virtual-machine:~/Documents/cpp$

```

Question 15:

```
/*
```

Question 15:

Create a C++ program that demonstrates the usage of the new and delete operators to dynamically allocate and deallocate memory for a single integer. Prompt the user to enter a value, allocate memory dynamically for the integer, assign the user input to it, display the value, and deallocate the memory.

```
*/
```

```
#include<iostream>
```

```
using namespace std;
```

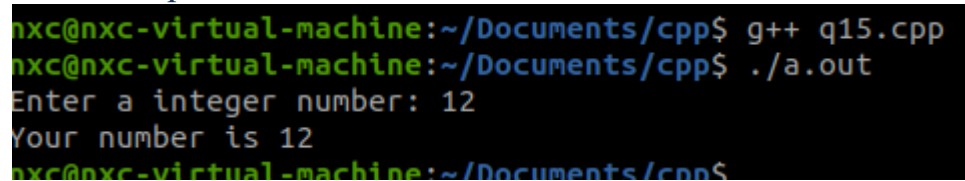
```
int main()
```

```

{
    int *p = new int;
    cout<<"Enter a integer number: ";
    cin>>*p;
    cout<<"Your number is "<<*p<<endl;
    delete p;
    return 0;
}

```

Result of question 15:



```

nxc@nxc-virtual-machine:~/Documents/cpp$ g++ q15.cpp
nxc@nxc-virtual-machine:~/Documents/cpp$ ./a.out
Enter a integer number: 12
Your number is 12
nxc@nxc-virtual-machine:~/Documents/cpp$

```

Question 16:

```
/*
```

Question 16:

Write a C++ program that demonstrates dynamic memory allocation and resizing using the realloc function. Prompt the user to enter the initial size of an array, allocate memory dynamically, allow the user to input values into the array, and then use the realloc function to resize the array based on user input.

```
*/
```

```
#include<iostream>
```

```
using namespace std;
```

```
int main()
```

```
{
```

```
    int q;
```

```
    cout<<"Quantity of input numbers is: ";
```

```
    cin>>q;
```

```
    int *dp = new int[q];
```

```
    cout<<"Enter values"<<endl;
```

```
for (int i=0; i < q; i++)
{
    cin>>*(dp+i);
}
cout<<"### All values:"<<endl;
for (int i=0; i < q; i++)
{
    cout<<*(dp+i)<<' ';
}
cout<<endl;
cout<<"### Resize - new quantity is: ";
cin>>q;
int *new_dp = (int *)realloc(dp, q);
cout<<"All values - After resize"<<endl;
for (int i=0; i < q; i++)
{
    cout<<*(new_dp+i)<<' ';
}
cout<<endl;
return 0;
}
```

Result of question 16:

```
nxc@nxc-virtual-machine:~/Documents/cpp$ g++ q16.cpp
nxc@nxc-virtual-machine:~/Documents/cpp$ ./a.out
Quantity of input numbers is: 5
Enter values
1
2
3
4
5
### All values:
1 2 3 4 5
### Resize - new quantity is: 3
All values - After resize
1 2 3
nxc@nxc-virtual-machine:~/Documents/cpp$ ./a.out
Quantity of input numbers is: 5
Enter values
1
2
3
4
5
### All values:
1 2 3 4 5
### Resize - new quantity is: 10
All values - After resize
1 2 3 4 5 0 59681 0 0 0
nxc@nxc-virtual-machine:~/Documents/cpp$
```

Question 17:

/\*

Question 17:

We have a sequence of data coming over a port. The method to get the data is

//=====

@brief: Rdata returns the port data.

Input: char\* data , A data array pointer is supplied.

Output: an integer number signifying the number of data bytes available.

//=====

int Rdata(char\* data) ;

Write a program which will allot a new array separate from the supplied pointer dynamically from the number of bytes (using keyword new) and copy the bytes from \*data to our newly created array.

```
*/
```

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

```
char arr[100];
```

```
int Rdata(char *data)
{
    int i = 0;
    while (*data)
    {
        arr[i] = *data;
        data++;
        i++;
    }
    return i;
}
```

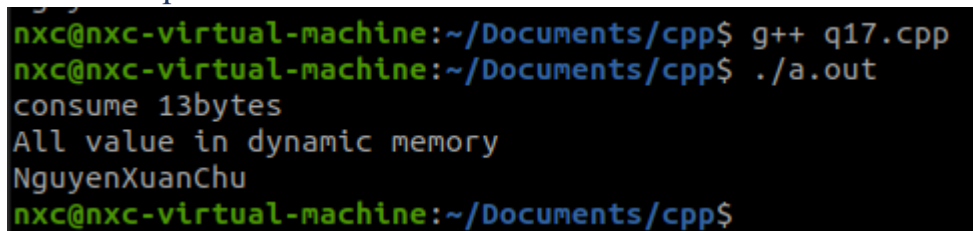
```
int main()
{
    char a[100] = "NguyenXuanChu";
    int byt = Rdata(a);
    // int byt = Rdata((char *)"NguyenXuanChu");
    cout<<"consume "<<byt<<"byte"<<endl;
    char *cp = new char[byt];
```

```

//write data from array to dynamic memory
for (int i=0; i < byt; i++)
{
    *(cp+i) = arr[i];
}
cout<<"All value in dynamic memory"<<endl;
for (int i=0; i < byt; i++)
{
    cout<<*(cp+i);
}
cout<<endl;
return 0;
}

```

Result of question 17:



```

nxc@nxc-virtual-machine:~/Documents/cpp$ g++ q17.cpp
nxc@nxc-virtual-machine:~/Documents/cpp$ ./a.out
consume 13bytes
All value in dynamic memory
NguyenXuanChu
nxc@nxc-virtual-machine:~/Documents/cpp$

```

Question 18:

/\*

Question 18:

Write a program which will get an input n from user in positive numbers, create an array of elements having n elements and values same as the index of the element,

dynamically. 0th index element value should be 0. The program should print the array and clear out the heap space before terminating.

\*/

```
#include<iostream>
```

```
using namespace std;
```

```

int main()
{
    int n;
    do
    {
        cout<<"Enter a positive number: ";
        cin>>n;
    } while (n<0);
    int *p = new int[n];
    for (int i=0; i<n; i++)
    {
        *(p+i) = i;
    }
    cout<<"Print array"<<endl;
    for (int i=0; i<n; i++)
    {
        cout<<*(p+i)<<' ';
    }
    cout<<endl;
    delete[] p;
    return 0;
}

```

Result of question 18:

```

nxc@nxc-virtual-machine:~/Documents/cpp$ g++ q18.cpp
nxc@nxc-virtual-machine:~/Documents/cpp$ ./a.out
Enter a positive number: 10
Print array
0 1 2 3 4 5 6 7 8 9
nxc@nxc-virtual-machine:~/Documents/cpp$

```

Question 19:

/\*

Question 19:

Write a program which will get two inputs from user, n and m. The program should



create two dynamic arrays to contain values ranging from 0 to n and 0 to m. Our program should print the two arrays and clear the memory before terminating.

```
*/
```

```
#include<iostream>
```

```
using namespace std;
```

```
int main()
```

```
{
```

```
    int n, m;
```

```
    do
```

```
    {
```

```
        cout<<"Enter a positive number: ";
```

```
        cin>>n;
```

```
    } while (n<0);
```

```
    do
```

```
    {
```

```
        cout<<"Enter a positive number: ";
```

```
        cin>>m;
```

```
    } while (m<0);
```

```
    //initial arrays
```

```
    int *np = new int[n];
```

```
    int *mp = new int[m];
```

```
    for (int i =0; i<n; i++)
```

```
    {
```

```
        *(np+i) = i;
```

```
    }
```

```
    for (int i =0; i<m; i++)
```

```
    {
```

```
        *(mp+i) = i;
```

```

}
//print all values
cout<<"All values of 2 arrays"<<endl;
cout<<"First array: ";
for (int i =0; i<n; i++)
{
    cout<<*(np+i)<<' ';
}
cout<<endl;
cout<<"Second array: ";
for (int i =0; i<m; i++)
{
    cout<<*(mp+i)<<' ';
}
cout<<endl;
//release memory
delete[] np;
delete[] mp;
return 0;
}

```

Result of question 19:

```

nxc@nxc-virtual-machine:~/Documents/cpp$ g++ q19.cpp
nxc@nxc-virtual-machine:~/Documents/cpp$ ./a.out
Enter a positive number: 5
Enter a positive number: 10
All values of 2 arrays
First array: 0 1 2 3 4
Second array: 0 1 2 3 4 5 6 7 8 9
nxc@nxc-virtual-machine:~/Documents/cpp$

```

Question 20:

/\*

Question 20:

Write a C++ program that employs templates to implement a generic function

called "swapValues" that swaps the values of two variables of any data type.

```
*/
```

```
#include<iostream>
```

```
using namespace std;
```

```
template <typename T>
```

```
void swapValues(T *a, T *b)
```

```
{
```

```
    T tmp;
```

```
    tmp = *a;
```

```
    *a = *b;
```

```
    *b = tmp;
```

```
}
```

```
int main()
```

```
{
```

```
    int n1 = 1;
```

```
    int n2 = 2;
```

```
    cout<<"Before swap: "<<n1<<' '<<n2<<endl;
```

```
    swapValues<int>(&n1, &n2);
```

```
    cout<<"After swap: "<<n1<<' '<<n2<<endl;
```

```
    return 0;
```

```
}
```

Result of question 20:

```
nxc@nxc-virtual-machine:~/Documents/cpp$ g++ q20.cpp
nxc@nxc-virtual-machine:~/Documents/cpp$ ./a.out
Before swap: 1 2
After swap: 2 1
nxc@nxc-virtual-machine:~/Documents/cpp$
```

Question 21:

```
/*
```

Question 21:

Write a c++ program that employs templates to implement an addition function using two values and returning a single result. The program should accept integer, float, long or double datatypes.

```
*/  
  
#include<iostream>  
using namespace std;  
  
template <typename T>  
T add(T a, T b)  
{  
    return a+b;  
}  
  
int main()  
{  
    int result1 = add<int>(2,3);  
    cout<<"addition = "<<result1<<endl;  
    float result2 = add<float>(2.132f,3.386f);  
    cout<<"addition = "<<result2<<endl;  
    long int result3 = add<long int>(200000,9999999999);  
    cout<<"addition = "<<result3<<endl;  
    double result4 = add<double>(2.3032,9.9215);  
    cout<<"addition = "<<result4<<endl;  
    return 0;  
}
```

Result of question 21:

```
nxc@nxc-virtual-machine:~/Documents/cpp$ g++ q21.cpp
nxc@nxc-virtual-machine:~/Documents/cpp$ ./a.out
addition = 5
addition = 5.518
addition = 10000199999
addition = 12.2247
nxc@nxc-virtual-machine:~/Documents/cpp$
```

Question 22:

/\*

Question 22:

Write a c++ program that compares between two numbers and returns the bigger number. Use templates so that the program is able to accept either integer or float values.

\*/

```
#include<iostream>
```

```
using namespace std;
```

```
template <typename T>
```

```
T compares(T a, T b)
```

```
{
```

```
    return a>b?a:b;
```

```
}
```

```
int main()
```

```
{
```

```
    cout<<"compare between 2 and 3"<<endl;
```

```
    cout<<"the bigger number is: "<<compares<int>(2,3)<<endl;
```

```
    cout<<"compare between 2.4 and 3.1"<<endl;
```

```
    cout<<"the bigger number is: "<<compares<float>(2.4,3.1)<<endl;
```

```
    return 0;
```

```
}
```

Result of question 22:

```
nxc@nxc-virtual-machine:~/Documents/cpp$ g++ q22.cpp
nxc@nxc-virtual-machine:~/Documents/cpp$ ./a.out
compare between 2 and 3
the bigger number is: 3
compare between 2.4 and 3.1
the bigger number is: 3.1
nxc@nxc-virtual-machine:~/Documents/cpp$
```

Question 23:

/\*

Question 23:

Write a c++ program which returns the average of 4 numbers.

We should be able to use either integer or float values with this program.

\*/

```
#include<iostream>
```

```
using namespace std;
```

```
template <typename T>
```

```
T average(T a, T b, T c, T d)
```

```
{
```

```
    return (a+b+c+d)/4;
```

```
}
```

```
int main()
```

```
{
```

```
    cout<<"The average of 1, 2, 3, 4 is "<<average<int>(1,2,3,4)<<endl;
```

```
    cout<<"The average of 1.1, 2.2, 3.3, 4.4 is  
"<<average<float>(1.1f,2.2f,3.3f,4.4f)<<endl;
```

```
    return 0;
```

```
}
```

Result of question 23:

```
nxc@nxc-virtual-machine:~/Documents/cpp$ g++ q23.cpp
nxc@nxc-virtual-machine:~/Documents/cpp$ ./a.out
The average of 1, 2, 3, 4 is 2
The average of 1.1, 2.2, 3.3, 4.4 is 2.75
nxc@nxc-virtual-machine:~/Documents/cpp$
```

Question 24:

/\*

Question 24:

Write a c++ programs which returns the sum of the elements for an array,

The user should be able to supply the array with n elements.

The elements may be either integer or float numbers.

\*/

```
#include<iostream>
```

```
using namespace std;
```

```
template<typename T>
```

```
T sum(T *arr, int count)
```

```
{
```

```
    T result = 0;
```

```
    for (int i=0; i<count; i++)
```

```
    {
```

```
        result += arr[i];
```

```
    }
```

```
    return result;
```

```
}
```

```
int main()
```

```
{
```

```
    int n,sel;
```

```
    cout<<"Enter quantity of numbers: ";
```

```

cin>>n;
cout<<"Choose the datatype\n1. Integer type\n2. Float type"<<endl;
do{
    cout<<"Your choice: ";
    cin>>sel;
} while(sel<1||sel>2);
int iarr[n];
float farr[n];
cout<<"Enter all "<<n<<" values"<<endl;
switch(sel)
{
    case 1:
        for (int i=0; i<n; i++)
        {
            cin>>iarr[i];
        }
        cout<<"sum is "<<sum<int>(iarr, n);
        break;
    case 2:
        for (int i=0; i<n; i++)
        {
            cin>>farr[i];
        }
        cout<<"sum is "<<sum<float>(farr, n);
        break;
}
cout<<endl;
return 0;
}

```



Result of question 24:

```
nxc@nxc-virtual-machine:~/Documents/cpp$ g++ q24.cpp
nxc@nxc-virtual-machine:~/Documents/cpp$ ./a.out
Enter quantity of numbers: 5
Choose the datatype
1. Integer type
2. Float type
Your choice: 1
Enter all 5 values
1 2 3 4 5
sum is 15
nxc@nxc-virtual-machine:~/Documents/cpp$ ./a.out
Enter quantity of numbers: 5
Choose the datatype
1. Integer type
2. Float type
Your choice: 2
Enter all 5 values
1.1 2.2 3.3 4.4 5.5
sum is 16.5
nxc@nxc-virtual-machine:~/Documents/cpp$
```

Question 25:

/\*

Question 25:

Write a C++ program that uses the STL algorithm library to sort an array of integers in ascending order. Prompt the user to enter the array elements and display the sorted array using the sort function.

\*/

```
#include<iostream>
```

```
#include<vector>
```

```
#include<algorithm>
```

```
using namespace std;
```

```
int main()
```

```
{
```

```
    int num;
```

```
    int tmp;
```

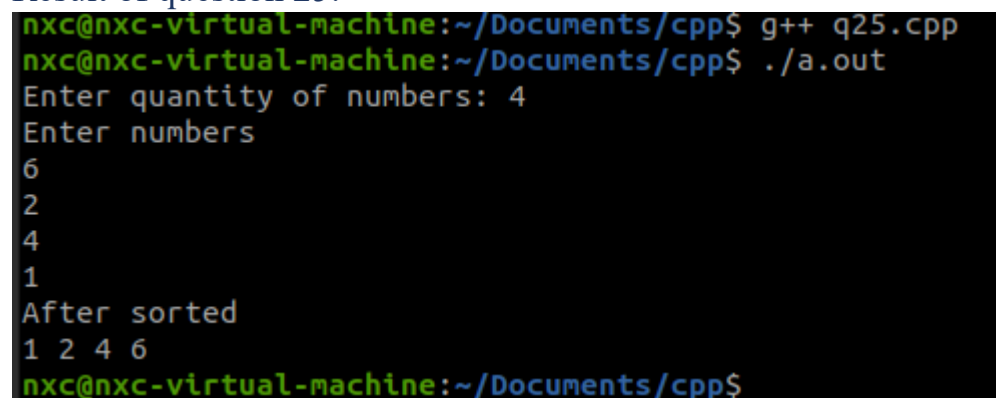
```
    vector <int> arr;
```

```

    cout<<"Enter quantity of numbers: ";
    cin>>num;
    cout<<"Enter numbers"<<endl;
    for (int i=0; i<num; i++)
    {
        cin>>tmp;
        arr.push_back(tmp);
    }
    sort(arr.begin(), arr.end());
    cout<<"After sorted"<<endl;
    for (int i=0; i<arr.size(); i++)
    {
        cout<<arr[i]<<" ";
    }
    cout<<endl;
    return 0;
}

```

Result of question 25:



```

nxc@nxc-virtual-machine:~/Documents/cpp$ g++ q25.cpp
nxc@nxc-virtual-machine:~/Documents/cpp$ ./a.out
Enter quantity of numbers: 4
Enter numbers
6
2
4
1
After sorted
1 2 4 6
nxc@nxc-virtual-machine:~/Documents/cpp$

```

Question 26:

/\*

Question 26:

We have a class of students. With name and roll no. Please create a base class to hold the details of any number of students. The program should ask the user for the number of students and the student details. The program should use the

vector container to keep all student information and iterate through them one by one printing all the student details in a legible and orderly manner.

```
*/
```

```
#include<iostream>
```

```
#include<vector>
```

```
#include<algorithm>
```

```
using namespace std;
```

```
class Student
```

```
{
```

```
    private:
```

```
    string name;
```

```
    int roll_no;
```

```
    public:
```

```
    void get_info()
```

```
    {
```

```
        cout<<"name: ";
```

```
        cin>>name;
```

```
        cout<<"roll number: ";
```

```
        cin>>roll_no;
```

```
    }
```

```
    void disp()
```

```
    {
```

```
        cout<<"name: "<<name<<endl;
```

```
        cout<<"roll number: "<<roll_no<<endl;
```

```
    }
```

```
};
```

```
int main()
```

```

{
    vector<Student> v;
    vector<Student>::iterator i;
    Student s;
    int n;

    cout<<"Number of student: ";
    cin>>n;
    for (int i=0; i<n; i++)
    {
        cout<<" Student "<<(i+1)<<endl;
        s.get_info();
        v.push_back(s);
    }
    cout<<"### All students ###"<<endl;
    i = v.begin();
    while(i != v.end())
    {
        (*i).disp();
        i++;
        cout<<endl;
    }
    return 0;
}

```

Result of question 26:

```
nxc@nxc-virtual-machine:~/Documents/cpp$ g++ q26.cpp
nxc@nxc-virtual-machine:~/Documents/cpp$ ./a.out
Number of student: 3
  Student 1
name: nguyen
roll number: 1
  Student 2
name: xuan
roll number: 2
  Student 3
name: chu
roll number: 3
### All students ###
name: nguyen
roll number: 1

name: xuan
roll number: 2

name: chu
roll number: 3

nxc@nxc-virtual-machine:~/Documents/cpp$
```

Question 27:

/\*

Question 27:

We have a class of employees with details such as name and Id no. Please create a base class for any number of employees and use a list container to contain all employee info. Our program should ask for total no of employees then each

detail and store them, also purse through the list from the end to print each employee detail in a legible and orderly manner.

\*/

```
#include<iostream>
```

```
#include<list>
```

```
#include<algorithm>
```

```
using namespace std;
```

```

class Employee
{
    private:
        string name;
        int id;
    public:
        void get_info()
        {
            // cout<<"  Get Employee Infomation"<<endl;
            cout<<"Name: ";
            cin>>name;
            cout<<"ID: ";
            cin>>id;
        }
        void disp()
        {
            // cout<<"  Employee Detail"<<endl;
            cout<<"Name: "<<name<<endl;
            cout<<"ID: "<<id<<endl;
        }
};

```

```

int main()
{
    int num;
    list<Employee> ls;
    list<Employee>::iterator it;
    Employee emp;
    cout<<"Enter number of employees: ";
    cin>>num;

```

```

for (int i=0; i<num; i++)
{
    cout<<"Fill in infomation of employee number "<<i<<endl;
    emp.get_info();
    ls.push_back(emp);
}
//display all employees
cout<<"All Employees Infomation"<<endl;
it = ls.begin();
while(it != ls.end())
{
    (*it).disp();
    it++;
}
return 0;
}

```

Result of question 27:

```

nxc@nxc-virtual-machine:~/Documents/cpp$ g++ q27.cpp
nxc@nxc-virtual-machine:~/Documents/cpp$ ./a.out
Enter number of employees: 3
Fill in infomation of employee number 0
Name: John
ID: 1
Fill in infomation of employee number 1
Name: Jack
ID: 2
Fill in infomation of employee number 2
Name: Janna
ID: 3
All Employees Infomation
Name: John
ID: 1
Name: Jack
ID: 2
Name: Janna
ID: 3
nxc@nxc-virtual-machine:~/Documents/cpp$

```

### Question 28:

/\*

Question 28:

Create a base class Shape with two private data members: width and height.

Implement a constructor to initialize these values and a function display () to print the width and height. Derive two classes Rectangle and Triangle from Shape.

Each derived class should have a specific function to calculate its area.

\*/

```
#include<iostream>
```

```
using namespace std;
```

```
class Shape
```

```
{
```

```
    private:
```

```
    int width;
```

```
    int height;
```

```
    public:
```

```
    //constructor
```

```
    Shape(int w, int h)
```

```
    {
```

```
        width = w;
```

```
        height = h;
```

```
    }
```

```
    //set_width set_height
```

```
    void set_width(int w)
```

```
    {
```

```
        width = w;
```

```
    }
```

```
    void set_height(int h)
```

```
    {
```

```
        height = h;
```



```

    }
    //get_width get_height
    int get_width()
    {
        return width;
    }
    int get_height()
    {
        return height;
    }
    void display()
    {
        cout<<"Width: "<<width<<endl;
        cout<<"Height: "<<height<<endl;
    }
};

class Rectangle: public Shape
{
    public:
    Rectangle(int width, int height):Shape(width, height)
    {

    }

    int area()
    {
        return get_width()*get_height();
    }
};

class Triangle: public Shape
{

```

```

public:
Triangle(int width, int height):Shape(width, height)
{

}

int area()
{
    return get_width()*get_height()/2;
}
};

int main()
{
    Rectangle rec(10,20);
    Triangle tr(10,20);

    rec.display();
    cout<<"Area of Rectangle = "<<rec.area()<<endl;

    tr.display();
    cout<<"Area of Triangle = "<<tr.area()<<endl;

    return 0;
}

```

Result of question 28:

```

nxc@nxc-virtual-machine:~/Documents/cpp$ g++ q28.cpp
nxc@nxc-virtual-machine:~/Documents/cpp$ ./a.out
Width: 10
Height: 20
Area of Rectangle = 200
Width: 10
Height: 20
Area of Triangle = 100
nxc@nxc-virtual-machine:~/Documents/cpp$ █

```

### Question 29:

/\*

Question 29:

Define a base class Animal with a virtual function sound(). Create two derived classes: Dog and Cat. Implement the sound() function in each derived class to produce a specific sound (e.g., barking for Dog and meowing for Cat). Demonstrate polymorphism by creating an array of pointers to the base class and assigning objects of derived classes to it. Iterate through the array, calling the sound() function for each object.

\*/

```
#include<iostream>
```

```
using namespace std;
```

```
class Animal
```

```
{
```

```
    public:
```

```
    virtual void sound()
```

```
    {
```

```
        cout<<"Sound of animal"<<endl;
```

```
    }
```

```
};
```

```
class Dog: public Animal
```

```
{
```

```
    public:
```

```
    void sound()
```

```
    {
```

```
        cout<<"Dog's sound: baw baw"<<endl;
```

```
    }
```

```
};
```

```
class Cat: public Animal
```

```
{
```

```

public:
void sound()
{
    cout<<"Cat's sound: meow meow"<<endl;
}
};

```

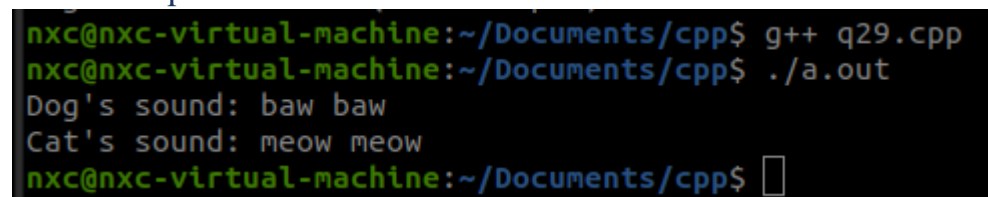
```

int main()
{
    // Animal *a1 = new Dog;
    // Animal *a2 = new Cat;
    Animal *a[] = {new Dog, new Cat};
    for (int i=0; i<2; i++)
    {
        a[i]->sound();
    }

    return 0;
}

```

Result of question 29:



```

nxc@nxc-virtual-machine:~/Documents/cpp$ g++ q29.cpp
nxc@nxc-virtual-machine:~/Documents/cpp$ ./a.out
Dog's sound: baw baw
Cat's sound: meow meow
nxc@nxc-virtual-machine:~/Documents/cpp$ 

```

Question 30:

```
/*
```

Question 30:

Create a base class Vehicle with a virtual function displayInfo(). Derive two classes: Car and Bike. Each derived class should have a constructor to initialize its attributes and override the displayInfo() function to print specific information about the vehicle. Use dynamic memory allocation to create an array of pointers

to the base class. Allow the user to input information for each vehicle, allocate memory, and then display the information using the displayInfo() function.

```
*/
```

```
#include<iostream>
```

```
using namespace std;
```

```
class Vehicle
```

```
{
```

```
    public:
```

```
    virtual void displayInfo()
```

```
    {
```

```
        cout<<"Vehicle information"<<endl;
```

```
    }
```

```
};
```

```
class Car: public Vehicle
```

```
{
```

```
    private:
```

```
    string brand;
```

```
    int price;
```

```
    public:
```

```
    Car(string b, int p)
```

```
    {
```

```
        brand = b;
```

```
        price = p;
```

```
    }
```

```
    void displayInfo() override
```

```
    {
```

```
        cout<<"Car Information"<<endl;
```

```
        cout<<"Brand: "<<brand<<endl;
```

```

        cout<<"Price: "<<price<<endl;
    }
};
class Bike: public Vehicle
{
    private:
        string brand;
        int price;
    public:
        Bike(string b, int p)
        {
            brand = b;
            price = p;
        }
        void displayInfo() override
        {
            cout<<"Bike Information"<<endl;
            cout<<"Brand: "<<brand<<endl;
            cout<<"Price: "<<price<<endl;
        }
};
int main()
{
    int num;
    int sel;
    string brand;
    int price;
    cout<<"Enter number of vehicle: ";
    cin>>num;

```

```

// int *ptr = new int[20];
Vehicle* *vptr = new Vehicle*[num];

cout<<"Fill in information"<<endl;

for (int i=0; i<num; i++)
{
    do
    {
        cout<<"1. Select car\n"
        <<"2. Select bike\n"
        <<"Choice: ";
        cin>>sel;
    } while (sel<1 && sel>2);
    switch(sel)
    {
        case 1:
            cout<<"Car brand: ";
            cin>>brand;
            cout<<"Car price: ";
            cin>>price;
            vptr[i] = new Car(brand, price);
            break;
        case 2:
            cout<<"Bike brand: ";
            cin>>brand;
            cout<<"Bike price: ";
            cin>>price;
            vptr[i] = new Bike(brand, price);
            break;
    }
}

```

```
        default:
            cout<<"Error | No choice"<<endl;
            break;
    }
}

//displayInfo
cout<<"### All Vehicles Information"<<endl;
for (int i=0; i<num; i++)
{
    vptr[i]->displayInfo();
}

//deallocate memory
delete[] vptr;

return 0;
}
```



Result of question 30:

```
nxc@nxc-virtual-machine:~/Documents/cpp$ g++ q30.cpp
nxc@nxc-virtual-machine:~/Documents/cpp$ ./a.out
Enter number of vehicle: 4
Fill in information
1. Select car
2. Select bike
Choice: 1
Car brand: toyota
Car price: 100
1. Select car
2. Select bike
Choice: 1
Car brand: vinfast
Car price: 90
1. Select car
2. Select bike
Choice: 2
Bike brand: yamaha
Bike price: 50
1. Select car
2. Select bike
Choice: 2
Bike brand: honda
Bike price: 40
### All Vehicles Information
Car Information
Brand: toyota
Price: 100
Car Information
Brand: vinfast
Price: 90
Bike Information
Brand: yamaha
Price: 50
Bike Information
Brand: honda
Price: 40
nxc@nxc-virtual-machine:~/Documents/cpp$
```