Matriculation Number:

Exercise 1.

(a) Implement a function `maximum` that takes a vector of `floats` as argument and returns the maximum of the elements in the vector. The vector shall be passed as a const reference, and for an empty vector 0 shall be returned and an error message shall be displayed.

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

float maximum(const vector<float> &v)
{
    if(v.size() == 0) {
        cout << "Error: vector is empty!" << endl;
        return 0;
    }

    vector<float>::const_iterator it = v.begin();
    float m = *it++;

    for(; it != v.end(); ++it)
        if(*it > m) m = *it;

    return m;
}
```

Matriculation Number:

(b) Implement a function `maxVector` that takes two vectors of `shorts` as arguments and returns the vector of maximums, i.e., the i-th element in the output vector shall be the maximum of the i-th elements in the input vectors.

If the two input vectors have different sizes, an error message shall be displayed and an empty vector shall be returned.

Example: Given the two vectors (1,2,5,6) and (2,-1,4,7), the result shall be = (2,2,5,7).

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

vector<short>
maxVector(const vector<short> &a, const vector<short> &b)
{
    if(a.size() != b.size()) {
        cout << "Error: vectors have different lengths!"
        << endl;
        return vector<short>();
    }

    vector<short> v(a.size());
    for(vector<short>::size_type i = 0; i < a.size(); ++i)
        v[i] = max(a[i],b[i]);

    return v;









 }
```

Matriculation Number:

Exercise 2.

Given a structure `Point` for representing points in 3D (see code below), extend the implementation as follows:

(a) Add a default constructor that initializes all coordinates to 0.

(b) Add a constructor for initializing the three coordinates.

(c) Overload the - operator for subtracting one point from another.

(d) Overload the * operator for multiplying a scalar value with a point: Given a `float s` and a point `p` with coordinates (x, y, z), then `s * p` is the point with coordinates $(s \cdot x, s \cdot y, s \cdot z)$.

```
struct Point {
    float x, y, z;

    Point() : x(0), y(0), z(0) { }
    Point(float xc, float yc, float zc) : x(xc), y(yc), z(zc) { }
};

Point operator-(const Point &p, const Point &q)
{
    return Point(p.x-q.x, p.y-q.y, p.z-q.z);
}

Point operator*(float s, const Point &p)
{
    return Point(s*p.x, s*p.y, s*p.z);
}
```

Matriculation Number:

Exercise 3.

Design a class hierarchy for persons (class `Person`), students (class `Student`), and master students (class `MasterStudent`), such that a person has a `name` (of type `string`), a student is a person with an additional `matric_number` (of type `int`), and a master student is a student with an additional `subject` (of type `string`).

Provide suitable constructors for the classes and make sure that all data members are private. Write a virtual member function `info` that prints all information (i.e., the data members) about a person and override this function in the derived classes such that also the additional information about students and master students is printed. When overriding `info`, call the `info` function of the base class first and then print the additional information.

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

class Person {
    string name;
public:
    Person(const string &n) : name(n) { }

    virtual void info() const {
        cout << "name = " << name << endl;
    }
};

class Student : public Person {
    int matric_number;
public:
    Student(const string &n, int mn)
        : Person(n), matric_number(mn) { }

    void info() const {
        Person::info();
        cout << "matr.nr = " << matric_number << endl;
    }
};
```

Matriculation Number:

```
class MasterStudent : public Student {
    string subject;
public:
    MasterStudent(const string &n, int mn, const string &s)
        : Student(n,mn), subject(s) { }

    void info() const {
        Student::info();
        cout << "subject = " << subject << endl;
    }
};
```

Matriculation Number:

Exercise 4.

(a) Read and understand the following program:

```
#include <iostream>

using namespace std;

int main()
{
  int x = 0, y = 0, z = 0;
  int &a = y;
  int *p1 = &x, *p2 = &y, *p3 = &z;

  *p1 = 100;  a = 20;
  cout << "x=" << x << ", y=" << y << ", z=" << z << endl; // CHECK1

  p3 = p1; p1 = &z;
  ++a; *p1 = *p3 + 5;
  cout << "x=" << x << ", y=" << y << ", z=" << z << endl; // CHECK2

  p1 = p3;
  a += *p2;
  *p1 -= *p3;
  cout << "x=" << x << ", y=" << y << ", z=" << z << endl; // CHECK3

  return 0;
}
```

Fill the table with the values of the variables x, y, and z at the three checkpoints.

	x	y	z
CHECK1	100	20	0
CHECK2	100	21	105
CHECK3	0	42	105

Matriculation Number:

(b) Read and understand the following program:

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

class C {
  static int counter;
  int c;
public:
  C() { c = ++counter; }
  int get_c() { return c; }

  virtual void id() { cout << "class C" << endl; }
  void mycount() { cout << "C: " << c << endl; }
};

int C::counter = 10;

class D : public C {
public:
  void id() { cout << "class D" << endl; }
  void mycount() { cout << "D: " << get_c() << endl; }
};

int main() {
  C c; D x;
  C *px = &x;

  x.id(); x.mycount();
  px->id(); px->mycount();

  return 0;
}
```

What is the output of the program?

```
class D
D: 12
class D
C: 12
```

Matriculation Number:

Room for Notes

Matriculation Number:

Room for Notes

Matriculation Number:

Room for Notes