**Aufgabe 1 (Strings und Vektoren).** **(4+4 Punkte)**

a) Geben Sie eine Funktion an, die als Argument einen String erwartet (`std::string`) und die von diesem String die Buchstaben mit geradem Index auf er Konsole ausgibt.

**Beispiel**: Für den String "`abcdef`" gibt die Funktion "`ace`" aus.

b) Geben sie eine Funktion an, die als Argument einen Vektor aus ganzen Zahlen erwartet und die das Produkt der Elemente zurückliefert.

**Aufgabe 2 (Fehlersuche).** **(6+6 Punkte)**

Betrachten Sie das folgende Programm:

```
include<iostream>
#include<string>

void foo(std::string s) {
    i=0;
    while s[i] != 'a' {
        i++;
    }
    while (i != 0) {
        std::cout << s[i] << '\n';
        i--
    }


int main() (
    foo("Dieses Programm macht dubiose Dinge!");
    return 42;
}
```

a) Das Programm hat 6 Fehler. Markieren Sie die Fehler.
**Für jede falsche Markierung wird Ihnen ein Punkt abgezogen.**

b) Geben Sie jeweils an, was falsch ist bzw. wie der Fehler korrigiert werden kann.

**Hinweis**: In jeder Zeile gibt es höchstens einen Fehler. **Ein Fehler liegt nur vor, wenn das Programm nicht compiliert werden kann.** Wenn eine Zeile nur eine Warnung produziert oder das Programm sich unerwartet verhält, liegt kein Fehler vor.

**Aufgabe 3 (Überladen von Funktionen).** **(5+2 Punkte)**

a) Überladen Sie den Operator `+` für den Datentyp `vector<int>`. D.h. für zwei `int`-Vektoren `v1` und `v2` soll `v1 + v2` die Verkettung von `v1` und `v2` sein.

b) Überladen Sie die folgende Funktion für den Datentyp `vector<int>`.

```
std::string verdoppeln(std::string s) { return s + s; }
```

**Aufgabe 4 (Klassen).** **(3+8 Punkte)**

a) Implementieren Sie den Konstruktor der folgenden Klasse, so dass er die Attribute der Klasse initialisiert.

```
class adresse {
private:
    std::string name;
    std::string strasse;
    std::string ort;
public:
    adresse(std::string n, std::string s, std::string o);
};
```

b) Definieren Sie eine Klasse `adressliste`, die eine Reihe von Adressen speichert. Die Klasse soll die folgenden Methoden haben:

- Eine Methode zum Hinzufügen einer neuen Adresse.
- Eine Methode, die als Argument einen Namen erwartet und die dazugehörige Adresse auf der Konsole ausgibt. Ist der Name nicht vorhanden, soll eine Fehlermeldung ausgegeben werden.

Erläutern Sie dabei ggf., welche Änderungen an `adresse` notwendig sind.

**Aufgabe 5 (Programmverständnis).** **(8 Punkte)**

Geben Sie die Ausgabe des folgenden Programms an:

```
#include<iostream>

int foo(int x, int y) {
    std::cout << x;
    return x + y;
}

void bar(int & k) {
    int y = k;
    for (char b=3;b>=-1;b--) {
        k = foo(y,b);
    }
}

int main() {
    int x = 1;
    while (x != 0) bar(x);
    return 0;
}
```

**Aufgabe 6 (Referenzen, Polymorphie).** **(8 Punkte)**

Geben Sie die Ausgabe des folgenden Programms an:

```
#include<iostream>
#include<string>

class name {
protected :
    std::string vorname = "Max";
    std::string nachname = "Mustermann";
public :
    void print() { std::cout << vorname << " " << nachname << "\n"; }
    virtual int length() { return vorname.size() + nachname.size() + 1; }
};

class firmenname : public name {
public :
    firmenname() { nachname = "AG"; }
    void print() { std::cout << nachname << "\n"; }
    virtual int length() { return nachname.size(); }
};


int main() {
    name n;
    firmenname f;
    name &rn = n;
    name * pf = &f;

    n.print();
    std::cout << n.length() << "\n";
    f.print();
    std::cout << f.length() << "\n";

    rn.print();
    std::cout << rn.length() << "\n";
    pf->print();
    std::cout << pf->length() << "\n";
}
```

**Aufgabe 7 (Templates).** **(6 Punkte)**

Geben Sie eine parametrisierte Funktion `replace` an, die einen `vector` `v` und zwei dazu passende Elemente `x` und `y` erwartet. Die Funktion soll einen Vektor zurückliefern, in dem jedes Vorkommen von `x` durch `y` ersetzt ist.

Die Funktion soll mit jedem beliebigen `vector` funktionieren.