Zeiger und Arrays

Aufgabe 2.1 In folgendem Code sehen Sie drei verschiedene Varianten, um eine Variable innerhalb einer Funktion zu erstellen, die (im Wesentlichen) einen Wert vom Typ `int` repräsentieren soll.

```
void f() {
  int a = 1;
  int* b = new int(2);
  std::unique_ptr<int> c = std::make_unique<int>(3);
}
```

Beantworten Sie für alle drei Varianten die folgenden Fragen:

- Welche Syntax verwendet man, um den repräsentierten Wert zum Beispiel auf `4` zu setzen?
- In welchem Teil des Speichers befindet sich der durch die Variable repräsentierte Wert?
- Wann und wie wird der für den repräsentierten Wert verwendete Speicher wieder freigegeben?

Aufgabe 2.2 Führen Sie folgendes Programmfragment in Ihrem Kopf aus. Welche Ausgabe würden Sie auf den einzelnen Programmzeilen erwarten?

```
1  int i1 = 1; int i2 = 2;
2  std::cout << i1 << " " << i2 << std::endl;
3
4  int* p1 = &i2; int* p2 = &i1;
5  std::cout << *p1 << " " << *p2 << std::endl;
6
7  int** pp1 = &p2; int** pp2 = &p1;
8  std::cout << **pp1 << " " << **pp2 << std::endl;
9
10 *p1 = 3; *p2 = 4;
11 std::cout << i1 << " " << i2 << std::endl;
12
13 *pp1 = &i2; *pp2 = &i1;
14 std::cout << *p1 << " " << *p2 << std::endl;
15
16 pp1 = &p1; pp2 = &p2;
17 std::cout << **pp1 << " " << **pp2 << std::endl;
```

Aufgabe 2.3 Implementieren Sie eine Funktion `max`, die ein C-Array mit Zahlen als Parameter bekommt und dann die grösste Zahl in diesem Array bestimmt. Etwas genauer ausgedrückt erwartet die Funktion an der Adresse `first` ein `int`-Array der Länge `length` und gibt die Adresse der grössten Zahl in diesem Array zurück. Testen Sie Ihre Funktion mit der bereits implementierten `main`-Funktion.

```
int* max(int* first, size_t length) {
  // Ihr Code kommt hier hin
}

int main() {
  int array[] = {4, 3, 1, 6, 7, 9, 0, 2, 5, 8};
  if(*max(array, 10) == 9) {
    std::cout << "Correct!" << std::endl;
  }
}
```