Um die Aufgaben zu bearbeiten und die eigenen Lösungen zu überprüfen, bietet es sich an, eine Funktion zu benutzen, die den Inhalt eines Arrays auf der Konsole ausgibt. Zum Beispiel folgende:

```
void printArray(int* a, int length){
    std::cout << "{ ";
    for (int i = 0; i < length; i++)
        std::cout << a[i] << " ";
    std::cout << "}" << std::endl;
}
```

Aufgabe 1.
Implementieren Sie eine Funktion `compareArray(int* a, int* b, int length)`, die zwei gleich lange Arrays `a` und `b` sowie ihre Länge `length` entgegennimmt. Sie soll `true` zurückgeben, wenn beide Arrays identisch sind, andernfalls `false`.

Aufgabe 2.
Was leistet die folgende Funktion? Betrachten Sie den Aufruf `psum(a,5)` für das Array `a = [5, 11, -2, 6, -8, 42]` und geben Sie für jede Iteration der `for`-Schleife den Inhalt von `result` an.

```
int* psum(int* a, int length){
  int* result = new int[length];
  result[0] = a[0];
  for (int i = 1; i < length; i++){
      result[i] = result[i-1] + a[i];
  }
  return result;
}
```

Aufgabe 3.
Implementieren Sie eine Funktion `zip(int* a, int* b, int length)`, die zwei gleich lange Arrays `a` und `b` sowie ihre Länge `length` entgegennimmt. Sie soll ein Array zurückgeben[1], das abwechselnd die Elemente von `a` und `b` enthält.
Beispiel: Ist `a = [1, 2, 3, 4, 5]` und `b = [32, 33, 34, 35, 36]`, so soll die Funktion das Array `[1, 32, 2, 33, 3, 34, 4, 35, 5, 36]` zurückliefern.

[1] Hiermit ist ein Pointer auf den Beginn des Arrays gemeint. Der Rückgabetyp ist also `int*`.

Aufgabe 4.

a) Implementieren Sie eine Funktion `count(int* a, int length, int x)`, die ein Array `a` sowie seine Länge `length` und eine weitere Zahl `x` entgegennimmt. Die Funktion soll zurückgeben, wie oft `x` in `a` vorkommt.
Beispiel: Ist `a = [32, 17, 1, -5, 8, 17]`, so soll `count(a,6,17)` das Ergebnis `2` liefern.

b) Implementieren Sie eine Funktion `filter(int* a, int length, int x)`, die ein Array `a` sowie seine Länge `length` und eine weitere Zahl `x` entgegennimmt. Die Funktion soll ein Array zurückgeben, das alle Elemente von `a` enthält, die nicht `x` sind (in der gleichen Reihenfolge).
Beispiel: Ist `a` wie oben, so soll `filter(a,6,17)` das Array `[32, 1, -5, 8]` zurückgeben.
Hinweis: Nutzen Sie die Funktion `count` aus Aufgabenteil a), um die Länge des resultierenden Arrays zu bestimmen.