Humboldt-Universität zu Berlin
Institut für Informatik
Prof. Dr. Verena Hafner
R. Lieck, H. Mellmann, Dr. W. Müller, M. Sänger

Berlin, den 20.01.2020

Übungen zur Vorlesung
Grundlagen der Programmierung

Übungsblatt 6

Abgabe: Die Abgabe Ihrer Lösung erfolgt über den Moodle-Kurs der Vorlesung (https://moodle.hu-berlin.de/course/view.php?id=90545). Nutzen Sie die im Abschnitt „Übung“ angegebene **Aufgabe 6.** Die Abgabe ist bis zum **03.02.2020** um **09:15 Uhr** möglich. **Die Abgabe erfolgt in Gruppen.** Bitte bilden Sie im Moodle-Gruppen! Bitte verwenden Sie **keine** (noch nicht in der Vorlesung eingeführten) Java-Bibliotheken.

Hinweis: Testen Sie Ihre Lösung bitte auf einem Rechner aus dem Informatik Computer-Pool z.B. `gruenau6.informatik.hu-berlin.de` (siehe auch Praktikumsordnung: Referenzplattform).

Ziel: Die Aufgaben des Übungsblatts sollen den Umgang mit Zahlensystemen, Bestimmung der Laufzeitkomplexität und Sortierverfahren vertiefen.

Aufgabe 1 (Zahlensysteme) — 5 Punkte

Wandeln Sie die folgenden Zahlendarstellungen um:

1) Basis 10 (Dezimal) ➔ Basis 17

2020_{10}

2) Basis 9 ➔ Basis 3

2020_{9}

3) Basis 10 (Dezimal) mit Vorzeichen ➔ **16-Bit Binärzahl im Zweierkomplement**

-2020_{10}

4) 3-Byte hexadezimalen Zweierkomplement ➔ **Dezimalzahl mit Vorzeichen**

$FFF81E_{16}$

5) 32-Bit Darstellung für `float` gemäß IEEE-754 ➔ entsprechende Dezimalzahl mit Vorzeichen und den notwendigen Kommastellen.

$1100000111001001000000000000000$

Geben Sie Ihre Lösungen als **UTF8-Textdatei A1.txt** ab. Ihre Antworten sollen dabei wie im folgenden Beispiel formatiert sein:

```
1: 42_17
2: 42_3
3: 123467890123456_2
4: +42
5: +4.2
```

Aufgabe 2 (Komplexität) 5 Punkte

Untersuchen Sie die Laufzeit-Komplexität folgender Java-Funktionen in Abhängigkeit von **n**. Bestimmen Sie die **kleinstmögliche** Komplexitätsklasse in der O-Notation.

1)

```java
public int f1(int n) {
    int sum = 0;
    for(int i = 0; i < n; i++) {
        sum += i;
    }
    return sum;
}
```

2)

```java
public int f2(int n) {
    int sum = 0;
    for(int i = 0; i < n; i++) {
        for(int j = 0; j < i; j++) {
            sum += i;
        }
    }
    return sum;
}
```

3)

```java
public int f3(int n) {
    int sum = 0;
    for(int i = 0; i < 1000; i++) {
        sum += n;
    }
    return sum;
}
```

4)

```java
public int f4(int n) {
    int sum = 0;
    for(int i = 0; i < n/2; i++) {
        sum += i;
    }
    return sum;
}
```

5)

```
public int f5(int n) {
    if (n <= 0) {
        return 0;
    }

    return f5(n/2) + n;
}
```

Geben Sie Ihre Lösungen als **UTF8-Textdatei A2.txt** ab. Ihre Antworten sollen dabei wie im folgenden Beispiel formatiert sein:

```
1: O(n)
2: O(n^2)
3: O(log(n))
4: O(n*log(n))
5: O(1)
```

Aufgabe 3 (Sortieren) 10 Punkte

Selection-Sort ist ein einfaches Sortierverfahren. Die grundlegende Idee hinter dem Algorithmus kann in folgenden Schritten zusammengefasst werden:

1. Am Anfang ist das gesamte Array unsortiert;
2. Wähle aus dem unsortierten Teil des Arrays das kleinste Element;
3. Vertausche das kleinste Element mit dem ersten Element des unsortierten Teils des Arrays (damit wird der unsortierte Teil um einen Element kleiner);
4. Wiederhole die Schritte 2 und 3 bis das gesamte Array sortiert ist;

Der folgende Pseudocode beschreibt den Selection-Sort Algorithmus:

```
prozedur selectionsort( A : Array sortierbarer Elemente )
  hoechsterIndex = elementanzahl( A ) - 1
  einfuegeIndex = 0
  wiederhole
      minPosition = einfuegeIndex
      für jeden idx von (einfuegeIndex + 1) bis hoechsterIndex wiederhole
          falls kleiner(A[ idx ], A[ minPosition ]) dann
              minPosition = idx
          ende falls
      ende für
      vertausche A[ minPosition ] und A[ einfuegeIndex ]
      einfuegeIndex = einfuegeIndex + 1
  solange einfuegeIndex < hoechsterIndex
prozedur ende
```

Implementieren Sie das Sortierverfahren **Selection-Sort** in Java, das ein Array von Strings sortiert. Zum Vergleich zweier Strings wird die Häufigkeit eines vorgegebenen Buchstaben verwendet. Nutzen Sie dafür die vorgegebene Vorlage **`SelectionSort.java`.**

- Vergleich zwischen Strings: Sei **c** ein Buchstabe (`char`). Dann bezeichnen wir einen `String s1` **kleiner** als ein `String s2`, falls der Buchstabe **c** in dem `String s2` häufiger auftritt.
- **Hinweis:** Haben mehrere Strings die gleiche Häufigkeit des vorgegebenen Buchstabens, so kann deren Reihenfolge beliebig gewählt werden.
- Der Buchstabe, nach dem sortiert werden soll, wird als Parameter an das Programm übergeben.
- Die zu sortierenden Strings werden über die Standardeingabe **zeilenweise** eingelesen (siehe Vorlage). D.h. jede Zeile enthält einen String. Eine Beispiel-Datei könnte zum Beispiel so aussehen:

```
Some say the world will end in fire,
Some say in ice.
From what I’ve tasted of desire
I hold with those who favor fire.
But if it had to perish twice,
I think I know enough of hate
To say that for destruction ice
Is also great
And would suffice.
```

(Der erste String hier wäre also: "`Some say the world will end in fire,`")

- Das sortierte Array soll in absteigender Reihenfolge auf die Konsole ausgegeben werden. Dabei soll jede Zeile wie folgt formatiert werden:

```
<N1>: <S1>
<N2>: <S2>
.
.
.
```

mit NX – Häufigkeit des gesuchten Buchstaben in dem jeweiligen String <SX>

Beispielaufruf:

```
java -cp ".;stdlib.jar" SelectionSort o < icefire.txt
```

Beispielausgabe:

```
4: I hold with those who favor fire.
3: I think I know enough of hate
3: To say that for destruction ice
2: Some say the world will end in fire,
2: From what I’ve tasted of desire
1: Some say in ice.
1: But if it had to perish twice,
1: Is also great
1: And would suffice.
```

Weitere Hinweise:

- Zum Einlesen der zu sortierenden Strings aus einer Datei wird die Klasse `StdIn` aus der Bibliothek `stdlib.jar` verwendet.
- Nutzen Sie für Ihre Lösung der Vorlage **`SelectionSort.java.`** Sie können die Vorlage nach Belieben verändern (also neue Funktionen einfügen etc.). Das Eingabe-/Ausgabe-Format muss aber eingehalten werden.
- In der Anlage finden Sie auch Beispieldateien die Eingabe, die Sie in Ihren Tests nutzen können: `test.txt, icefire.txt, faust.txt`

Denken Sie daran, Ihr Programmquellcode zu kommentieren und auf dem Referenzsystem zu testen. Laden Sie die Datei **`SelectionSort.java`** rechtzeitig bei Moodle hoch.