Xiaotian Yu
Matrikelnummer：3562309
E-mail：yuxiaotian2009@126.com

Nuo chen
Matrikelnummer：3561481
E-mail：2212806679@qq.com

Yuzhe Shao
Matrikelnummer：3559723
E-mail：1013804202@qq.com

Aufgabe 1

(a) *(4 Punkte)* Nutzen Sie das in den Vorlesungsfolien/-aufzeichnungen vorgestellten Schema zur *sequenziellen* Suche und wenden Sie diese auf die zwei gegebenen Listen an. Führen Sie dabei die einzelnen Schritte auf, damit deutlich wird, ob die *sequenzielle* Suche möglich ist und wie diese schrittweise abläuft.

(b) *(3 Punkte)* Nutzen Sie das in den Vorlesungsfolien/-aufzeichnungen vorgestellten Schema zur *binären* Suche und wenden Sie diese ebenfalls auf die zwei gegebenen Listen an. Führen Sie dabei die einzelnen Schritte auf, damit deutlich wird ob die *binäre* Suche möglich ist und wie diese schrittweise abläuft.

Gesuchtes Element: 11

1	23	47	43	68	12	11	73	81	76	54	21	23	90	90
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14

Gesuchtes Element: 48

12	13	23	47	48	68	69	73	81	83	84	85	87	90	94
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14

a)

```
for i := 0 to n – 1 do
        if F[i] = k then return i if
        if F[i] >k then return NO_KEY fi
od;
return NO_KEY;
```

List 1

The element we are looking for is 11.

We start from F[0], the element of f[0] is 1, it's smaller than 11.

Then we try the next element F[1]. F[1] = 23 ,it's bigger than 11

So we get the return : NO_KEY

List 2

The element we are looking for is 48

We start from F[0], the element of f[0] is 12, it's smaller than 48.

Then we try the next element F[1]. F[1] = 13 ,it's also smaller than 48

We repeat the above steps, and we get that F[2] and F[3] are smaller than 48

At last we get the goal element F[4] = 48. And we get the return 4.

b)

```
u:=0 ; o := n-1;
while u <= o do
        m:= ( u + o ) / 2;
        if F[m] = k then
                return m
        else if k < F[m] then o := m – 1;
                else u := m + 1;
        if
od;
return NO_KEY
```

List 1:

Cycle 1:

u = 0, o = 14

m = (0 + 14) / 2 = 7

F[7] = 73, 11<73 ,so o= 6

Cycle 2

m = (0 +6) / 2 = 3

F[3] = 43 , 11 < 43 . so o = 2

Cycle 3

m = (0 +2) / 2 = 1

F[1] = 23,11 < 23 , so o = 0

Cycle 4

m = (0 +0)/2 = 0

F[0] = 1; 11>1, so u = 1

u>o, return NO_KEY

List 2

Cycle 1:

u = 0, o = 14

m = (0 + 14) / 2 = 7

F[7] = 73, 48 < 73 ,so o = 6

Cycle 2

m = (0 +6) / 2 = 3

F[3] = 47 ,48 > 47 . so u = m +1 = 4

Cycle 3

m = (4 + 6)/2= 5

F[5] = 68, 48 <68, so o = 5-1 = 4

Cycle 4

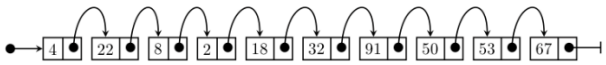
m = (4 + 4)/ 2 = 4

F[4] = 48

Return 4

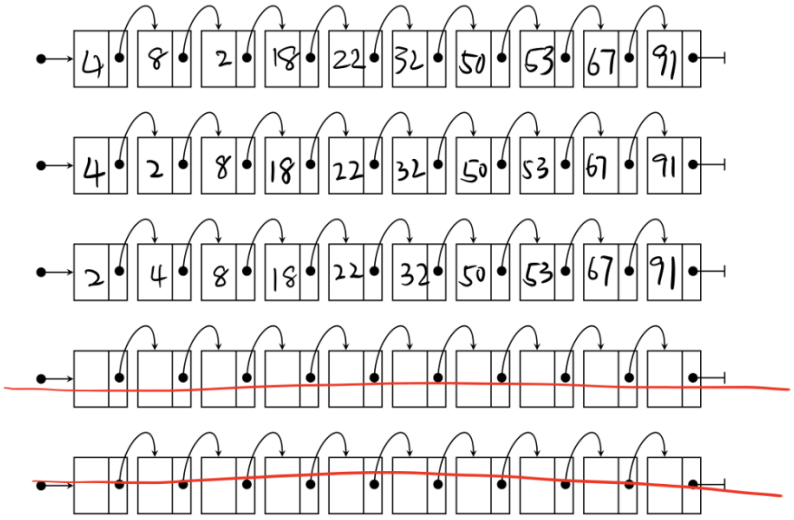
Aufgabe 3 BubbleSort *[Punkte: 4]*

Im Folgenden ist eine verkettete Liste mit Zahlenwerten abgebildet:



Die Liste soll mittels BubbleSort aufsteigend von vorne nach hinten sortiert werden.

(a) *(3 Punkte)* Tragen Sie im Folgenden ein, wie die Liste nach den einzelnen Durchläufen der äußeren Schleife von BubbleSort aussieht. Gehen Sie davon aus, dass der Algorithmus terminiert, sobald die Liste sortiert ist. Streichen Sie eventuell nicht benötigte Zeilen aus.



(b) *(1 Punkte)* Der BubbleSort-Algorithmus ließe sich auch so implementieren, dass die Liste von hinten nach vorne durchlaufen wird.

Was spricht bei der gezeigten Listenstruktur dagegen, den Algorithmus auf diese Weise zu implementieren? Begründen Sie in einem Satz.

Aufgabe 4 Sortierverfahren Komplexität *[Punkte: 3]*

Im Folgenden sind die schnellsten Sortierverfahren für unterschiedliche Anwendungsfälle gesucht. Markieren Sie im Folgenden für jeden Fall, welches Sortierverfahren aus der Vorlesung die gegebenen Daten am schnellsten (aufsteigend) sortiert. Gibt es mehrere gleich schnelle Verfahren, dann markieren Sie alle. Begründen Sie außerdem ihre Auswahl und geben Sie dabei zu jedem Anwendungsfall an, welche Komplexität das ausgewählte Verfahren besitzt.

Fall 1: Gegeben sind absteigend sortierte Daten

☐ QuickSort ☐ SelectionSort ☐ BubbleSort ☒ MergeSort ☐ InsertionSort

Begründung/Komplexität: $O(n \log n)$

Fall 2: Gegeben sind diese sortierten Daten => [5, 6, 8, 9, 11, 23, 32, 34, 58, 59, 123, 233, 436, 543]

☐ QuickSort ☐ SelectionSort ☒ BubbleSort ☐ MergeSort ☒ InsertionSort

Begründung/Komplexität: $O(n)$

Fall 3: Gegeben sind unsortierte („chaotische“) Daten

☒ QuickSort ☐ SelectionSort ☐ BubbleSort ☒ MergeSort ☐ InsertionSort

Begründung/Komplexität: $O(n \log n)$