Hochschule **Rosenheim**
University of Applied Sciences

Prüfung	**WS 2015/16**
Studiengang:	INF-B
Fach:	Grundlagen der Informatik 2
Prüfer:	Prof. Dr. J. Schmidt
Prüfung:	8.2.2016

90 Minuten. Hilfsmittel: alle Unterlagen, Taschenrechner, **kein** Laptop, Handy, u.ä.
Insgesamt sind 90 Punkte zu erreichen. Die Punktzahl gibt damit auch einen Anhaltspunkt für die Bearbeitungszeit.
Sollten Ihrer Meinung nach Angaben in der Aufgabenbeschreibung fehlen oder falsch sein, machen Sie sinnvolle Annahmen und dokumentieren Sie diese.
Die Seiten dürfen nicht getrennt werden.
Konzeptpapier muss (mit Namen versehen) mit abgegeben werden.

Aufgabe	1	2	3	4	5	6	7	8	9	ges.
Punkte										

Note:

Name: ______________________________ **Matrikelnr.:**________________

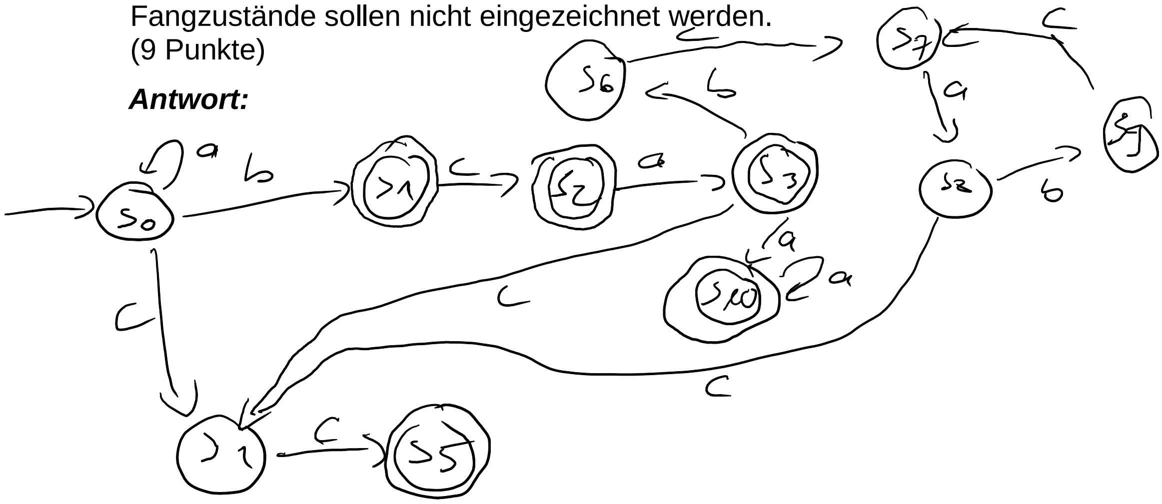
Aufgabe 1: Endliche Automaten (19 Punkte)

a) Zeichnen Sie das Übergangsdiagramm eines erkennenden endlichen Automaten, der folgende Sprache L über dem Alphabet T = {a, b, c} akzeptiert:

$L = \{ a^n (bca)^m cc \mid n, m \in \mathbb{N}_0 \} \cup \{ a^n b \mid n \in \mathbb{N}_0 \} \cup \{ a^n bca^m \mid n, m \in \mathbb{N}_0 \}$

Fangzustände sollen nicht eingezeichnet werden.
(9 Punkte)

Antwort:



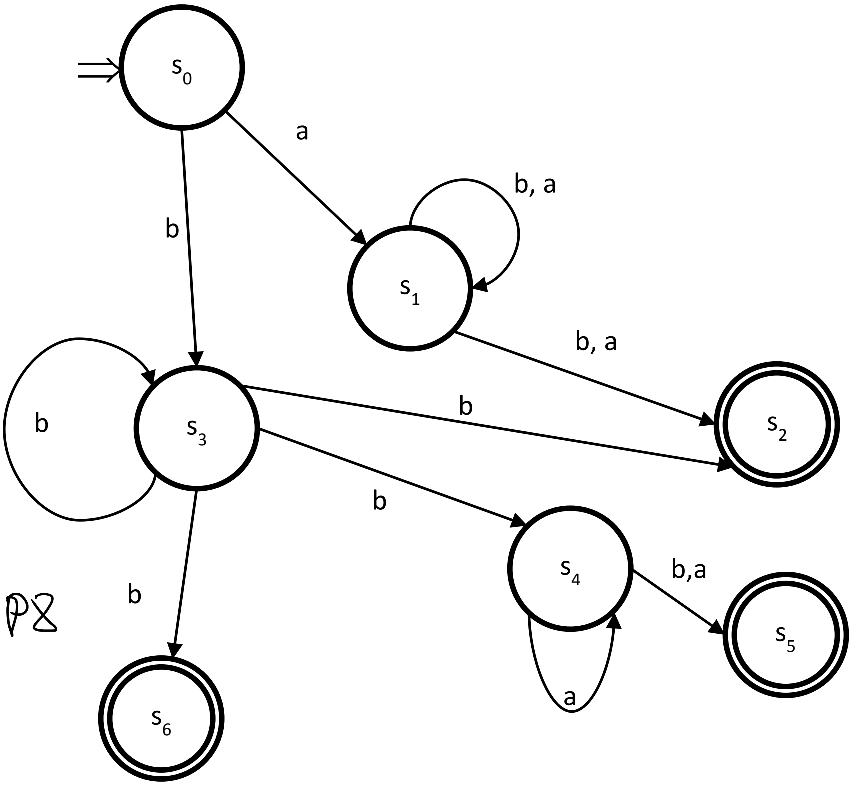
b) Geben Sie für den unten stehenden nichtdeterministischen endlichen Automaten die Übergangstabelle eines äquivalenten deterministischen Automaten an, der die gleiche Sprache akzeptiert.

- Verwenden Sie hierfür die Potenzmengenkonstruktion nach Rabin/Scott
- Eine Umbenennung der Zustandsmengen muss **nicht** durchgeführt werden
- Geben Sie die **Anfangs- und Endzustände** des neuen Automaten an

(10 Punkte)

A = P1

E = P4; P5; P6; P7; P8



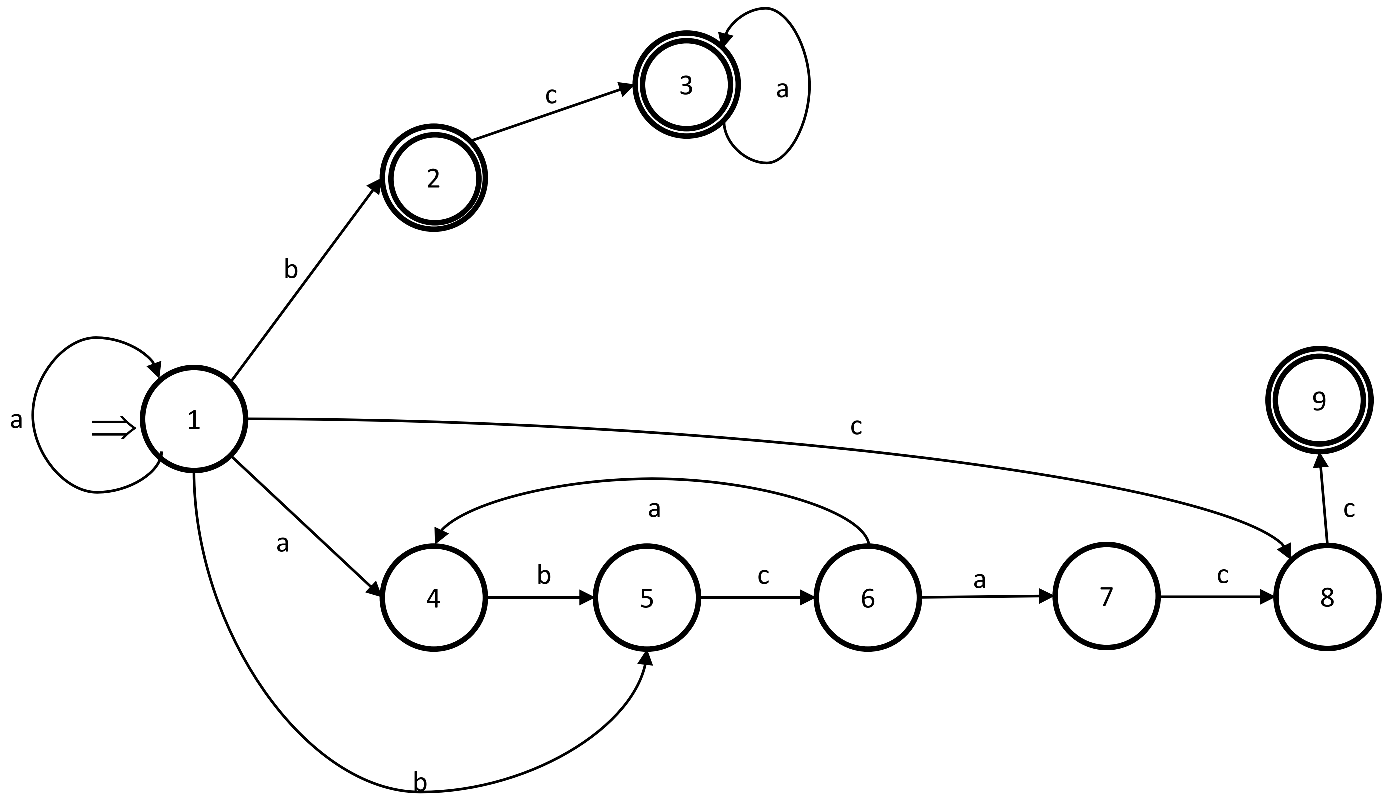
Antwort:

	$\{s_0\}$ P1	$\{s_1\}$ P2	$\{s_3\}$ P3	$\{s_1, s_2\}$ P4
a	$\{s_1\}$ P2	$\{s_1, s_2\}$ P4	—	$\{s_1, s_2\}$ P4
b	$\{s_3\}$ P3	$\{s_1, s_2\}$ P4	$\{s_2, s_3, s_4, s_6\}$ P5	$\{s_1, s_2\}$ P4

	$\{s_2, s_3, s_4, s_6\}$ P5	$\{s_4, s_5\}$ P6	$\{s_2, s_3, s_4, s_5, s_6\}$ P7	$\{s_5\}$ P8
a	$\{s_4, s_5\}$ P6	$\{s_4, s_5\}$ P6	$\{s_4, s_5\}$ P6	—
b	$\{s_2, s_3, s_4, s_5, s_6\}$ P7	$\{s_5\}$ P8	$\{s_2, s_3, s_4, s_5, s_6\}$ P7	—

Lösung

a)



b) Tabelle:

	{0}	{1}	{3}	{1, 2}	{2, 3, 4, 6}	{4, 5}	{2, 3, 4, 5, 6}	{5}
a	{1}	{1, 2}	-	{1, 2}	{4, 5}	{4, 5}	{4, 5}	-
b	{3}	{1, 2}	{2, 3, 4, 6}	{1, 2}	{2, 3, 4, 5, 6}	{5}	{2, 3, 4, 5, 6}	-

Anfangszustand: {0}

Endzustände: {1, 2}, {2, 3, 4, 6}, {2, 3, 4, 5, 6}, {4, 5}, {5}

Aufgabe 2: Grammatiken (8 Punkte)

Gegeben ist folgende Grammatik (Startsymbol Z, Terminalsymbole T = {a, b, c, d}:

$Z \rightarrow aZb \mid cZZd \mid cZaZ \mid ab \mid cd$

a) Von welchem Typ der Chomsky-Hierarchie ist diese Grammatik? Schränken Sie den Typ so weit wie möglich ein, begründen Sie Ihre Antwort.

Antwort: Typ 2, weil links nur ein nicht Terminal steht, dass es rechts nicht linear ist

b) Bringen Sie die Grammatik in Chomsky-Normalform. Die Schritte der Entstehung müssen erkennbar sein.

Antwort:

Lösung

a) Typ 2: kontextfrei, Produktionen haben die Form: $A \rightarrow u$ mit $A \in S$ und $u \in V^+$

b) Vorgehen:

- jedes Terminalsymb. erhält eine neue Variable
  $A \rightarrow a$, $B \rightarrow b$, $C \rightarrow c$, $D \rightarrow d$
  $Z \rightarrow AZB \mid CZZD \mid CZAZ \mid AB \mid CD$
- ersetzte Regeln mit mehr als 2 Variablen auf rechter Seite
  $A \rightarrow a$, $B \rightarrow b$, $C \rightarrow c$, $D \rightarrow d$
  $Z \rightarrow AE \mid CF \mid CH \mid AB \mid CD$
  $E \rightarrow ZB$,
  $F \rightarrow ZG$
  $G \rightarrow ZD$
  $H \rightarrow ZI$
  $I \rightarrow AZ$

Aufgabe 3: Verschiedenes (17 Punkte)

Welche der folgenden Aussagen sind richtig bzw. falsch? Kreuzen Sie das entsprechende Feld an. Falsche Antworten geben Punktabzug (wird nicht auf andere Aufgaben übertragen).

Aussage	**richtig**	**falsch**
Ein Algorithmus mit Zeitkomplexität O(n log n) ist für jede Größe der Eingangsdaten schneller als einer mit $O(2^n)$	X	
$n - 2n^2 + ½ n^3 = O(n)$		X
$10000n + 100n^2 + 2n^3 = O(n^5)$		X
$10000n + 100n^2 + 2n^3 = O(n^3)$	X	
Die Ackermann-Funktion ist WHILE-berechenbar, aber nicht total		X
Das Halteproblem ist für alle primitiv-rekursiven Probleme berechenbar		X
Ein Problem aus P ist mit einer nichtdeterministischen Turing-Maschine in polynomieller Zeit lösbar	X	
Falls P = NP gilt, dann gibt es einen polynomiellen Algorithmus zur exakten Lösung des SAT Problems		
Das Wortproblem für Typ-2 Sprachen ist entscheidbar		
Nichtdeterministische Kellerautomaten können mehr Probleme berechnen als deterministische Kellerautomaten	X	
Für die Berechnung des Wertes der Busy-Beaver Funktion an der Stelle 3 bb(3) kann eine Turing Maschine konstruiert werden, damit ist die Funktion bb(x) WHILE-berechenbar		
Mit einem CYK-Parser lässt sich das Wortproblem für Typ 3 Sprachen lösen	X	
Mit einer Turing Maschine lässt sich das Wortproblem für Typ 0 Sprachen lösen		X
Ein Greedy-Algorithmus findet immer die optimale Lösung für ein gegebenes Problem		
Das größte und kleinste Element aus einem unsortierten Feld mit natürlichen Zahlen zu finden geht mit Aufwand O(n)	X	
Jede LOOP-berechenbare Funktion ist auch GOTO-berechenbar	X	
Jede reguläre Grammatik lässt sich durch einen endlichen Automaten darstellen, der die gleiche Sprache akzeptiert		X

Lösung

Aussage	**richtig**	**falsch**
Ein Algorithmus mit Zeitkomplexität O(n log n) ist für jede Größe der Eingangsdaten schneller als einer mit $O(2^n)$		x
$n - 2n^2 + ½ n^3 = O(n)$		x
$10000n + 100n^2 + 2n^3 = O(n^5)$	x	
$10000n + 100n^2 + 2n^3 = O(n^3)$	x	
Die Ackermann-Funktion ist WHILE-berechenbar, aber nicht total		x
Das Halteproblem ist für alle primitiv-rekursiven Probleme berechenbar	x	
Ein Problem aus P ist mit einer nichtdeterministischen Turing-Maschine in polynomieller Zeit lösbar	x	
Falls P = NP gilt, dann gibt es einen polynomiellen Algorithmus zur exakten Lösung des SAT Problems	x	
Das Wortproblem für Typ-2 Sprachen ist entscheidbar	x	
Nichtdeterministische Kellerautomaten können mehr Probleme berechnen als deterministische Kellerautomaten	x	
Für die Berechnung des Wertes der Busy-Beaver Funktion an der Stelle 3 bb(3) kann eine Turing Maschine konstruiert werden, damit ist die Funktion bb(x) WHILE-berechenbar		x
Mit einem CYK-Parser lässt sich das Wortproblem für Typ 3 Sprachen lösen	x	
Mit einer Turing Maschine lässt sich das Wortproblem für Typ 0 Sprachen lösen		x
Ein Greedy-Algorithmus findet immer die optimale Lösung für ein gegebenes Problem		x
Das größte und kleinste Element aus einem unsortierten Feld mit natürlichen Zahlen zu finden geht mit Aufwand O(n)	x	
Jede LOOP-berechenbare Funktion ist auch GOTO-berechenbar	x	
Jede reguläre Grammatik lässt sich durch einen endlichen Automaten darstellen, der die gleiche Sprache akzeptiert	x	

Aufgabe 4: Berechenbarkeit (5 Punkte)

Zeigen Sie, dass die Funktion $f(n) = \binom{n}{2}$ für $n \geq 2$ primitiv rekursiv ist. Zusätzlich zur Definition der primitiven Rekursion dürfen Sie verwenden, dass die folgenden Funktionen ebenfalls primitiv rekursiv sind:

- Multiplikation: $m(x, y) = xy$
- Division: $d(x, y) = \frac{x}{y}$
- Vorgänger: $v(x) = x - 1$

$$\boxed{f(n) = \binom{n}{2} = \frac{n!}{2(n-2)!} = \frac{1}{2}\, n(n-1)}$$

Antwort:

Lösung

$f(2) = 1$

$$f(n+1) = \frac{(n+1)!}{2!\,(n-1)!} = \frac{1}{2}(n+1)n = \frac{1}{2}(n+1)n\frac{(n-1)}{(n-1)} = \frac{1}{2}n(n-1)\frac{(n+1)}{(n-1)} = f(n)\frac{(n+1)}{(n-1)} =$$

$m(f(n), d(s(n), v(n)))$

Aufgabe 5: Komplexität (6 Punkte)

Man kann nachweisen, dass für alle n mit $n \geq 4$ die Abschätzung gilt: $2^n < n! < n^n$

Welche der folgenden Beziehungen lassen sich daraus ableiten?

Falsche Antworten geben Punktabzug (wird nicht auf andere Aufgaben übertragen).

a) $2^n = O(n!)$ ✓	b) $2^n = \Omega(n^n)$ ✗	c) $2^n = \Theta(n^n)$ ✗
d) $2^n = O(2^n)$ ✓	e) $2^n = \Omega(2^n)$ ✓	f) $2^n = \Theta(2^n)$ ✓
g) $n! = O(2^n)$ ✗	h) $n! = \Omega(n^n)$ ✗	i) $n! = \Theta(n!)$ ✓
j) $n^n = O(2^n)$ ✗	k) $n^n = \Omega(n!)$ ✗	l) $n^n = \Theta(n!)$ ✗

Antwort:

Lösung

a, d, e, f, i, k

Aufgabe 6: Pseudo-Zufallszahlen (4 Punkte)

Mit der Formel $x_{n+1} = (a\, x_n + c) \bmod m$ lassen sich ganzzahlige Zufallszahlen erzeugen.

Es sei nun: a = 3, c = 9, m = 16

a) Berechnen Sie die ersten zwei Zufallszahlen x_1 und x_2 beginnend mit Startwert $x_0 = 1$

b) Ist garantiert, dass sich mit dieser Parameterwahl die maximal mögliche Periodenlänge ergibt? Begründen Sie Ihre Antwort.

Antwort:

Lösung

a) x0 = 1, x1 = 3*1 + 9 = 12, x2 = 3*12 + 9 = 45 = 13 (mod 16)

b) ggT(16, 9) = ggT(9, 7) = ggT(7, 2) = ggT(2,1) = 1 → keine gemeinsamen Primfaktoren
a – 1 = 2, m ist Vielfaches von 4 → da 2 kein Vielfaches von 4 → nicht max. Länge

Aufgabe 7: Komplexität (9 Punkte)

a) Sie stellen fest, dass ein Problem der Größe n nach dem Prinzip „Teile und Herrsche“ so in 16 Teilprobleme zerlegt werden kann, dass diese jeweils nur die Größe n/4 haben. Der Aufwand für die Kombination der Teillösungen zur Gesamtlösung sei in der Größenordnung O(n). Geben Sie für diesen Fall die resultierende Zeitkomplexität an.

Antwort:

b) Bestimmen Sie die Zeitkomplexität des folgenden Codeausschnitts in O-Notation. Die Variable n bezeichnet die zu verarbeitende Datenmenge.

- Geben Sie zunächst in der rechten Spalte für jede Anweisungszeile die Komplexität an (bei Logarithmen mit Basis), und bestimmen Sie anschließend die Gesamtkomplexität.
- Vereinfachen Sie die Gesamtkomplexität so weit wie möglich

Code	Zeitkomplexität
`for( i = n; i >= 0; i = i - 1 )`	
`{`	
`  int k = 3;`	
`  while( k > i )`	
`  {`	
`    int x = 2;`	
`    for(j = 1; j < n - 3; j = j * 2 )`	
`      x = x + 2;`	
`    for(j = n ; j > 2; j = j - 1 )`	
`    {`	
`      x = x + n;`	
`      x = x - 1;`	
`    }`	
`    k = k / 4;`	
`  }`	
`}`	

Gesamtkomplexität:

Lösung

a) $T(n) = a\,T(n/b) + \Theta(n^k)$
a = 16, b = 4, k = 1 → 16 > 4^1 → O(n^log_b a) = O(n^log_4 16) = $O(n^2)$

b)

Code	Zeitkomplexität
`for( i = n; i >= 0; i = i - 1 )`	O(n)
`{`	
`  int k = 3;`	O(1)
`  while( k > i )`	O(1)
`  {`	
`    int x = 2;`	O(1)
`    for(j = 1; j < n - 3; j = j * 2 )`	$O(\log_2 n)$
`      x = x + 2;`	O(1)
`    for(j = n ; j > 2; j = j - 1 )`	O(n)
`    {`	
`      x = x + n;`	O(1)
`      x = x - 1;`	O(1)
`    }`	
`    k = k / 4;`	O(1)
`  }`	
`}`	

Gesamtkomplexität:

$O(n) * (O(1) + O(1)*(O(1) + O(\log_2 n) + O(n) + O(1)))$

$= O(n) * (O(1) + O(1)*O(n)) = O(n) * O(n)$

$= \mathbf{O(n^2)}$

Aufgabe 8: Wortproblem (14 Punkte)

Gegeben ist folgende kontextfreie Grammatik (Startsymbol Z, Terminalsymbole T = {a, b}):

Z → AB | BC
A → BA | a
B → CC | b
C → AB | a

Prüfen Sie mit Hilfe des CYK-Algorithmus, ob das Wort aabbaaba Teil der durch die Grammatik definierten Sprache ist. Füllen Sie hierzu die unten stehende Tabelle aus:

a	a	b	b	a	a	b	a

Bitte kreuzen Sie an:
Das Wort ist Teil der Sprache: ☐ ja ☐ nein

Begründung:

Lösung

Wort ist Teil der Sprache, da im untersten Feld das Startsymbol entsteht

a	a	b	b	a	a	b	a
A, C	A, C	B	B	A, C	A, C	B	A, C
B	Z, C		A, Z	B	Z, C	A, Z	
B		A		B	B		
				Z, C, A			
A			Z, A, C				
		A, Z, C					
	B						
Z, C, A							

Aufgabe 9: Pumping Theorem (8 Punkte)

Zeigen Sie mit Hilfe des Pumping Theorems, dass die folgende Sprache **nicht kontextfrei** ist:

$L = \{a^i b^k c^k d^k \mid i, k \in \mathbb{N}_0\}$

Antwort:

Lösung

- sei L eine kontextfreie Sprache
- dann gibt es eine Konstante n, so dass sich jedes Wort $w \in L$, mit $|w| > n$ zerlegen lässt in $w = uvxyz$ mit
- $|vxy| \leq n$
- $|vy| \geq 1$
- $|z|, |x|$ beliebig (also auch 0)
- Es gilt dann: $u\, v^i\, x\, y^i\, z \in L$, für alle i = 0, 1, 2, …
-
- wähle $w = b^n c^n d^n$
- vxy enthält
  (1) entweder nur b
  (2) oder b und c
  (3) oder nur c
  (4) oder c und d
  (5) oder nur d
  aber nie alle drei Buchstaben zusammen
- Fall
  (1) beim Pumpen entstehen nur neue b
  (2) beim Pumpen entstehen nur neue b/c
  (3) beim Pumpen entstehen nur neue c
  (4) beim Pumpen entstehen nur neue c/d
  (5) beim Pumpen entstehen nur neue d
  in jedem Fall ist die Anzahl b/c/d anschließend unterschiedlich

voraussichtlicher Notenschlüssel:

0-36: 5,0	37 – 45: 4,0	46 – 50: 3,7	51 - 54: 3,3	55 - 59: 3,0	60 - 63: 2,7
	64 – 68: 2,3	69 – 72: 2,0	73 – 77: 1,7	78 – 81: 1,3	82 - 90: 1,0