"Klausur Informatik "

Muster für Übungen/Praktika

Wintersemester

1. Algorithmierung (9 Punkte)

   Gegeben sei eine endliche Folge von n ganzen Zahlen (n>1).
   Entwerfen und implementieren Sie ein C++-Programm mit folgender Funktionalität:
   Definieren Sie zunächst ein Feld konstanter Länge (z.B. 500). Der Nutzer des Programms soll dann eine Zahl eingeben, die die tatsächliche Zahl der einzugebenden Zahlen repräsentiert. Diese Zahl muss $\leq$ der definierten Feldlänge sein. Lesen Sie die Folge von der Standardeingabe in das Feld ein und zerlegen Sie die Folge in monotone Teilfolgen. Geben Sie die Teilfolgen auf der Standardausgabe aus. Bei jeder Zustandsänderung (steigend - fallend bzw. fallend - steigend) ist eine neue Zeile zu beginnen.
   Hinweis 1: Aufeinanderfolgende gleiche Zahlen ändern den Zustand nicht.
   Hinweis 2: Sie können davon ausgehen, dass die Zeilenlänge in einem Zustand nicht überschritten wird.
   Beispiel:

Eingabefolge:	1	1	2	4	7	3	2	2	1	6	5	4	1	4	6
Ausgabefolge:	1	1	2	4	7										
	3	2	2	1											
	6														
	5	4	1												
	4	6													

2. Rekursive Funktionen (8 Punkte)

   Die Funktion *mc* ist wie folgt definiert:

$$mc(n) = \begin{cases} n - 10 & \text{für n} > 100 \\ mc(mc(n + 11)) & \text{sonst} \end{cases}$$

   Diese Funktion wurde von *J. McCarthy* gefunden.

   a. Schreiben Sie eine C++-Funktion

   ```
   int mccarthy(int n)
   ```

die rekursiv den Wert der Funktion mc für ein $n \in N$ berechnet.

b. Schreiben Sie eine Funktion **int main()**, die die These, dass die Funktion mc im Intervall von $1 \leq n \leq 101$ immer das gleiche Ergebnis liefert, überprüft. Wenn die These zutrifft, soll zusätzlich ausgegeben werden, welches Ergebnis berechnet wird.

3. Strukturen, Funktionen (8 Punkte)

Gegeben sei eine Struktur **Student**

```
struct Student {
   char name[40];
   char vorname[20];
   float noten[50];
   int matrikelnummer;
};
```

Aus dieser Struktur wird ein Feld **studenten** aufgebaut:

```
Student studenten [1000];
```

welches global verfügbar ist. Das Feld sei bereits mit Werten belegt. Wenn weniger als 1000 Studenten enthalten sind, ist im ersten freien Eintrag im Strukturelement **matrikelnummer** der Wert -1 eingetragen. Für jeden Studenten existiert ein Feld noten, welches mit den bisher vergebenen Noten gefüllt ist. Die Zuordnung des Faches zu einem bestimmten Index sei fest vorgegeben. Für jedes Fach ist die erhaltene Note oder der Wert 0 eingetragen, wenn die Prüfung im betreffenden Fach noch nicht abgelegt wurde. Entwerfen und implementieren Sie eine C++-Funktion **get_durchschnitt**:

```
float get_durchschnitt(int m_nummer);
```

in der das Feld **studenten** sequentiell nach einem Element durchsucht wird, welches die matrikelnummer hat, die dem Parameter **m_nummer** entspricht. Die Funktion liefert als Ergebnis

- den Wert -1, wenn ein Student mit der Matrikelnummer **m_nummer** nicht existiert,
- den Wert 0, wenn der Student mit der Matrikelnummer **m_nummer** existiert und keine Prüfungen absolviert hat, sonst
- den Durchschnitt aller vorhandenen Noten des Studenten mit der Matrikelnummer **m_nummer**.

Es ist davon auszugehen, dass jede Matrikelnummer nur einmal auftritt.
Demonstrieren Sie die Verwendung der Funktion **get_durchschnitt** in einer

C++-Funktion main() zur Entscheidung darüber, welche Bewertung ein Student gemäß ECTS erhält (die ECTS-Bewertung soll auf dem Bildschirm erscheinen):

ECTS-Grad	Bezeichnung	dt. Durchschnitt
A	excellent	$1.0 \leq$ durchschnitt ≤ 1.5
B	very good	$1.5 <$ durchschnitt ≤ 2.0
C	good	$2.0 <$ durchschnitt ≤ 3.0
D	satisfactory	$3.0 <$ durchschnitt ≤ 3.5
E	sufficient	$3.5 <$ durchschnitt ≤ 4.0
FX/F	fail	$4.0 <$ durchschnitt ≤ 5.0

4. Matrixoperationen, Algorithmierung (15 Punkte)

Das Spiel des Lebens (engl. Conway's Game of Life) ist ein vom Mathematiker John Horton Conway 1970 entworfenes System. Das Spielfeld ist in Zeilen und Spalten unterteilt und im Idealfall unendlich groß. Jedes Gitterquadrat kann einen von zwei Zuständen einnehmen, welche oft als lebendig und tot bezeichnet werden. Zunächst wird eine Anfangsgeneration von lebenden Zellen auf dem Spielfeld platziert. Jede lebende oder tote Zelle hat auf diesem Spielfeld genau acht Nachbarzellen, die berücksichtigt werden. Da ein reales Spielfeld immer einen Rand hat, muss das Verhalten dort festgelegt werden. Man kann sich den Rand zum Beispiel durch Zellen belegt denken, die immer tot bleiben.

n	n	n
n	x	n
n	n	n

Abb.: 1: Das Gitterquadrat **x** hat die mit **n** gekennzeichneten Nachbarn.

Die nächste Generation ergibt sich durch die Befolgung einfacher Regeln. Betrachtet wird im folgenden die in der Mitte einer 3 x 3 Matrix stehende Zelle, lebendige Zellen werden mit **1** gekennzeichnet, tote mit **0**. Zellen, die mit einem **?** gekennzeichnet sind, haben einen noch nicht bestimmten eindeutigen Inhalt. Bei einer Implementierung müssen natürlich auch diese Zellen betrachtet werden.

– Eine tote Zelle mit genau drei lebenden Nachbarn wird in der Folgegeneration neu geboren.

0	0	0
0	**0**	0
1	1	1

vorher

?	?	?
?	**1**	?
?	?	?

nachher

– Lebende Zellen mit weniger als zwei lebenden Nachbarn sterben in der Folgegeneration an Einsamkeit.

0	0	0
0	**1**	0
0	0	1

vorher

?	?	?
?	**0**	?
?	?	?

nachher

– Eine lebende Zelle mit zwei oder drei lebenden Nachbarn bleibt in der Folgegeneration lebend.

0	0	0
1	**1**	0
0	1	1

vorher

?	?	?
?	**1**	?
?	?	?

nachher

– Lebende Zellen mit mehr als drei lebenden Nachbarn sterben in der Folgegeneration an Überbevölkerung.

1	1	1
1	**1**	1
1	1	1

vorher

?	0	?
0	**0**	0
?	0	?

nachher

Gegeben sei eine 25 x 25 Integer-Matrix, deren Elemente die Zellen repräsentieren. Die Matrix sei mit einer Anfangsgeneration von lebenden Zellen belegt. Jede Zelle hat auf diesem Spielfeld genau acht Nachbarzellen, die eindeutig den Zustand der Zelle in der nächsten Generation bestimmen. Der Rand der Matrix (Zeilen 0 und 24, Spalten 0 und 24) ist mit Nullen belegt. Entwerfen und implementieren Sie ein C++-Programm, das drei Lebenszyklen der Teilmatrix, die durch die Elemente mit den Indizes

– 1,1
– 1,23
– 23,1
– 23,23

begrenzt wird, berechnet. Geben Sie den Zustand der Matrix nach jedem Lebenszyklus auf dem Bildschirm aus. Jedes Element sollte dabei mit einer Breite von 3 Zeichen linksbündig ausgegeben werden.
Hinweis: Berechnen Sie einen Zustand n+1 aus dem Zustand n in einer Hilfsmatrix und kopieren deren Werte dann in die eigentliche Matrix zurück.