Live-Programmiersitzung am 05./07./08.01.2026

Aufgabe 1 Bausparguthaben (25 Punkte)

In dieser Aufgabe ist ein Programm zu entwickeln, das das Guthaben auf einem Bausparkonto berechnet.

a) Erstellen Sie ein C-Programm, das den Benutzer zunächst auffordert, eine positive natürliche Zahl ($B \geq 5000$) für die Höhe der erwünschten Bausparsumme einzugeben (in Euro). Die Aufforderung zur Eingabe soll so lange wiederholt werden, bis der Benutzer eine Zahl größer oder gleich 5000 eingegeben hat.

b) Als Nächstes ist vom Benutzer der Wert e abzufragen, der angeben soll, wie viel der Kunde pro Jahr in seinen Bausparvertrag einzahlt. Die Höhe der jährlichen Einzahlungen darf $\frac{1}{20}$ der Bausparsumme nicht überschreiten und muss größer als null sein. Die Aufforderung zur Eingabe soll so lange wiederholt werden, bis der Benutzer eine Zahl e mit $0 < e \leq \frac{B}{20}$ eingegeben hat.

c) Das Bausparguthaben wird jährlich mit einem Zinssatz von 3% verzinst. Berechnen Sie unter Verwendung der aus der Vorlesung bekannten Kontrollstruktur `do-while`-Schleife (nicht-abweisende Schleife), wie hoch das Bausparguthaben zum Ende eines Jahres ist und geben Sie dies für alle Jahre aus, bis die Bausparsumme B erreicht ist. Tätigen Sie zusätzlich eine Ausgabe in den Jahren, in denen der Bausparvertrag "zuteilungsreif" ist (dies ist der Fall, wenn 50% der Bausparsumme erreicht worden sind).
Hinweis: Rechnen Sie mit zwei Stellen Genauigkeit (Cent-Beträge).

Beispielhafter Programmablauf:

```
Geben Sie die anvisierte Bausparsumme (mind. 5000 Euro) ein: 4000
Geben Sie die anvisierte Bausparsumme (mind. 5000 Euro) ein: 20000
Geben Sie den jährlichen Einzahlungsbetrag (größer null und nicht größer als 1/20 der
Bausparsumme) ein: 0
Geben Sie den jährlichen Einzahlungsbetrag (größer null und nicht größer als 1/20 der
Bausparsumme) ein: 1100
Geben Sie den jährlichen Einzahlungsbetrag (größer null und nicht größer als 1/20 der
Bausparsumme) ein: 1000
Bausparguthaben nach 1 Jahren: 1030.00
Bausparguthaben nach 2 Jahren: 2090.90
Bausparguthaben nach 3 Jahren: 3183.63
Bausparguthaben nach 4 Jahren: 4309.14
Bausparguthaben nach 5 Jahren: 5468.41
Bausparguthaben nach 6 Jahren: 6662.46
Bausparguthaben nach 7 Jahren: 7892.34
Bausparguthaben nach 8 Jahren: 9159.11
Bausparguthaben nach 9 Jahren: 10463.88 (Bausparsumme ist zuteilungsreif!)
Bausparguthaben nach 10 Jahren: 11807.80 (Bausparsumme ist zuteilungsreif!)
Bausparguthaben nach 11 Jahren: 13192.03 (Bausparsumme ist zuteilungsreif!)
Bausparguthaben nach 12 Jahren: 14617.79 (Bausparsumme ist zuteilungsreif!)
Bausparguthaben nach 13 Jahren: 16086.32 (Bausparsumme ist zuteilungsreif!)
Bausparguthaben nach 14 Jahren: 17598.91 (Bausparsumme ist zuteilungsreif!)
Bausparguthaben nach 15 Jahren: 19156.88 (Bausparsumme ist zuteilungsreif!)
Bausparguthaben nach 16 Jahren: 20761.59 (Bausparsumme ist zuteilungsreif!)
Bausparsumme erreicht!
```

Aufgabe 2 Primzahlfelder (40 Punkte)

a) Um zu überprüfen, ob eine gegebene Zahl eine Primzahl ist, sollen Sie eine Funktion namens

```
int prim_test(int z)
```

schreiben, die als Parameter z die zu überprüfende Zahl entgegennimmt und die als Rückgabewert eine `1` zurückliefern soll, sofern es sich um eine Primzahl handelt und `0` anderenfalls.
Hinweis: Sie müssen bei Ihrer Implementierung nicht auf algorithmische Effizienz achten. Außerdem darf hierbei davon ausgegangen werden, dass die übergebene Zahl positiv und größer als eins ist.

b) Schreiben Sie ein C-Programm, das im Hauptprogramm zunächst vom Benutzer 6 Mal eine Zahl, die größer als eins ist, einliest. Sollte der Benutzer eine Zahl n mit $n \leq 1$ eingegeben haben, so ist die Eingabe solange zu wiederholen, bis der Benutzer eine Zahl größer eins eingegeben hat.

Erzeugen Sie zwei statische Felder der Größe 6 (für bis zu 6 Integer-Werte): Das Feld `prims` soll Primzahlen speichern, das Feld `non_prims` soll Zahlen speichern, die keine Primzahlen sind. Ihr Programm soll bei jeder eingegebenen Zahl überprüfen, ob eine Primzahl eingegeben wurde (hierfür ist die in a) implementierte Funktion aufzurufen), und diese Zahl entsprechend dem Feld `prims` oder dem Feld `non_prims` hinzufügen. Überlegen Sie, wie Sie sich merken können, wie viele Primzahlen-/Nicht-Primzahlen Sie bereits in das jeweilige Feld eingetragen haben.

c) Geben Sie anschließend die Inhalte der beiden Felder vollständig aus, so dass zunächst alle eingegebenen Primzahlen und anschließend alle eingegebenen Nicht-Primzahlen ausgegeben werden.

d) Ermitteln Sie und geben Sie aus, ob die Summe der eingelesenen Primzahlen oder die Summe der eingelesenen Nicht-Primzahlen größer ist (oder ob beide gleich sind). Hierfür ist eine Funktion zu schreiben und zu verwenden, die als Parameter ein Feld sowie dessen Größe übergeben bekommt und als Rückgabewert die Summe der Elemente des Feldes zurückliefert.

Beispiel:

```
Geben Sie bitte die 1. Zahl ein: 3
Geben Sie bitte die 2. Zahl ein: 1
Geben Sie bitte die 2. Zahl ein: 0
Geben Sie bitte die 2. Zahl ein: -11
Geben Sie bitte die 2. Zahl ein: 5
Geben Sie bitte die 3. Zahl ein: 9
Geben Sie bitte die 4. Zahl ein: 9
Geben Sie bitte die 5. Zahl ein: 7
Geben Sie bitte die 6. Zahl ein: 6
Primzahlen: 3 5 7
Nicht-Primzahlen: 9 9 6
Die Summe der Primzahlen (15) ist kleiner als die Summe der Nicht-Primzahlen (24).
```

Aufgabe 3 Happy Birthday! (25 Punkte)

Sie möchten Ihre/n beste/n Freund/in mit einem nerdigen Geburtstagsglückwunsch überraschen.
Im Verzeichnis `vorlagen/` ist Ihnen die aus den Übungen bekannte Bibliothek `advanced_console_output.[ch]` zur Verfügung gestellt. Schreiben Sie unter Verwendung dieser Bibliothek ein C-Programm, das Ihren Geburtstagsglückwunsch wie folgt in der Konsole darstellt.

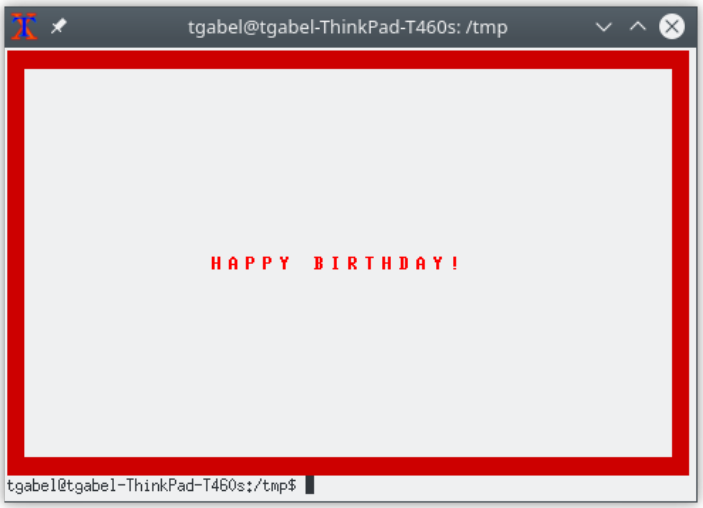

Abbildung 1: Beachten Sie: Dieses Aufgabenblatt ist in Grautönen gedruckt. Im hier dargestellten Bild sind der Rahmen sowie der Text in der Mitte rot. Zusätzlich blinkt der Text in der Mitte.

Ihr Programm soll

- den Bildschirm löschen,
- den Bildschirm rot umrahmen, d.h.
  - obere Zeile rot,
  - am linken und rechten Konsolenrand je zwei rote Leerzeichen,
  - zweitunterste (nicht die unterste!) Zeile komplett rot,
- den Text `H A P P Y  B I R T H D A Y !` darstellen, und zwar
  - in roter Schriftfarbe,
  - blinkend,
  - fett gedruckt,
  - möglichst exakt mittig ausgerichtet in der Konsole.
- zum Ende den Cursor an den Anfang der untersten Zeile im Terminal versetzen, Vorder-/Hintergrundfarbe wieder auf "default" stellen und alle Textattribute (kein bold/blinking mehr!) wieder zurücksetzen.

Orientieren Sie sich an der Darstellung in Abbildung 1.

Aufgabe 4 Pseudo-Aktienkurs (45 Punkte)

Ziel der Aufgabe ist es, ein Programm zu entwickeln, das den Verlauf eines fiktiven Aktienkurses simuliert und darstellt. Hierbei ist insbesondere von der aus den Übungen bekannten Bibliothek Advanced Console Output Gebrauch zu machen, die Sie im Verzeichnis `vorlagen/` vorfinden.
Abbildung 2 zeigt einen Screenshot des Programmes. Eine lauffähige Version (Binary) der Lösung finden Sie auf den Klausurrechnern unter `vorlagen/`.

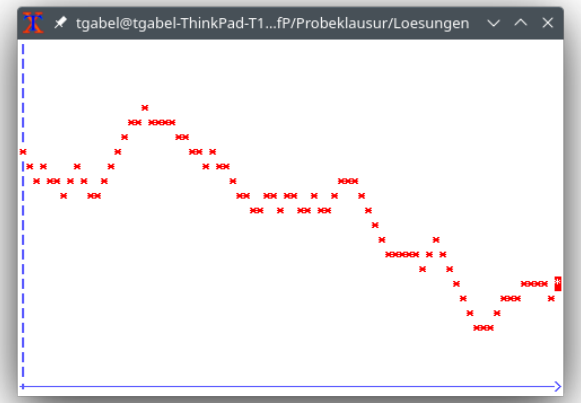

Abbildung 2: Beispielhafter Pseudo-Aktienkurs. Da dieser Klausurbogen in Grautönen gedruckt ist, ist nicht erkennbar, dass die Achsen in blau und der Kursverlauf in rot dargestellt ist.

a) Initialisieren Sie den Zufallszahlengenerator systemzeitbasiert, löschen Sie den Bildschirm und ermitteln Sie die Breite (b) und Höhe (h) des Terminals, in dem Sie das Programm laufen lassen.

b) Zeichnen Sie ein aus x- und y-Achse bestehendes Koordinatensystem, indem Sie für die x-Achse in der untersten Bildschirmzeile durchgängig das Zeichen `'-'` (Bindestrich) verwenden und ganz rechts ein `'>'` (Größerzeichen) platzieren. Die y-Achse soll mit dem Zeichen `'|'` in der ersten Spalte des Terminals dargestellt werden. Für den Koordinatenursprung ist das `'+'` (Pluszeichen) zu verwenden. Beide Achsen sollen mit blauer Textfarbe dargestellt werden.

c) Der Kursverlauf soll in roter Textfarbe, **fett** und mit dem Zeichen `'*'` (Stern) dargestellt werden.

d) Der Kursverlauf soll wie folgt errechnet und dargestellt werden:

- Der Startwert des Aktienkurses (bei $x = 0$, d.h. in Spalte 1 der Textausgabe) soll zufällig bestimmt werden.
- Eine Zählschleife ist zu verwenden, um den Aktienkurs von links nach rechts (also von der ersten Spalte bis zur letzten) zu berechnen und darzustellen.

- In jedem Schritt entlang der x-Achse (also von Spalte zu Spalte) soll der Kurs mit je einer Wahrscheinlichkeit von $\frac{1}{3}$ entweder konstant bleiben, um eins wachsen oder um eins sinken.
  Hinweis: Mit "eins" ist hier eine Einheit entlang der y-Achse (also eine Zeile) gemeint.
- Der Aktienkurs ist nach unten durch $y = 0$ (unterste Bildschirmzeile) und nach oben durch die Höhe des Bildschirms (also durch die oberste Zeile des Terminals) zu begrenzen: Sollte der Aktienkurs diese Grenzen unter-/überschreiten, wird er auf jene Minimal-/Maximalwerte zurückgesetzt.
- Nach dem Zeichnen jedes `'*'` soll unter Verwendung der Funktion `sleep(1)` eine Sekunde gewartet sowie der Ausgabepuffer der Standardausgabe mit der Funktion `fflush(stdout)` geleert (und damit zur Ausgabe auf den Bildschirm gebracht) werden.

e) Nachdem der Kursverlauf gezeichnet worden ist, soll das Programm in einer abweisenden Endlosschleife hängen bleiben (so dass es im Terminal nur mit Ctrl+C abgebrochen werden kann).