technische universität
dortmund

Klausurdeckblatt

Matrikel – Nr.:

Bitte tragen Sie Ihre Matrikelnummer und Ihren Namen in die dafür vorgesehenen Felder ein. Bitte in deutlicher Handschrift mit einem schwarzen Stift (nicht Bleistift).
Das Feld mit dem **Barcode ist unbedingt frei zu lassen.**
Danke.

Vorname:

Nachname:

Der Bereich unterhalb dieser Linie kann von der Fakultät frei gestaltet werden.

Einführung in die Programmierung (Roman Kalkreuth)

P R O B E K L A U S U R

Studiengang: ☐ ET/IT ☐ IKT ☐ WiMa ☐ _____

Übungsgruppe: _____ (Nummer oder Termin & Name des Tutors)

Hinweise:
Sie haben zur Bearbeitung dieser Probeklausur 90 Minuten zur Verfügung. Hilfsmittel jeglicher Art (mit Ausnahme dokumentenechter Schreibgeräte) sind nicht erlaubt. Die Aufsicht gibt Ihnen keine Hilfestellung bei der Lösung dieser Aufgaben und wird entsprechende Anfragen von Ihrer Seite nicht beantworten (mit Ausnahme von sprachlichen Verständnisfragen).

Für das „Bestehen“ dieser Probeklausur sind mindestens **12 Punkte** erforderlich. Für die Aufgaben der Klausur können Sie die folgenden Punkte erhalten:

Aufgabe	Erreichbare Punkte	Erzielte Punkte
1	10	
2	10	
3	10	
Summe	30	

Bevor Sie mit der Bearbeitung der Aufgaben beginnen, tragen Sie auf allen Blättern Ihren Namen und Ihre Matrikelnummer ein. Sie sollten sich zuerst alle Aufgaben durchlesen und dabei auf die von Ihnen geforderten Ergebnisse achten, bevor Sie versuchen, die Aufgaben zu lösen. **Viel Erfolg!**

Aufgabe 1: ***Programmverständnis*** [10 Punkte]

a) Geben Sie für das unten angegebene und in drei Blöcken unterteilte Programm an, welche Ausgabe erzeugt wird.

a_1) *[4 Punkte]*

```
#include <iostream>
using namespace std;

void machwas(int& x) {
  x = x / 4;
}

int main() {
  char feld[] = "number";
  int x = 3;
  for (unsigned int i = 0; feld[i] != 0; i++) {
    switch (feld[i]) {
    case 'u':
      x = 0;
      break;
    case 'b':
    case 'm':
      x = x - 4;
    case 'e':
      x = x + 4;
      break;
    case 'n':
      cout << x;
      break;
    default:
      cout << x;
      machwas(x);
      break;
    }
    cout << x;
  }
  cout << endl;
```

Antwort: _____

a_2) *[1 Punkt]*

```
// char feld[] = "number"; ist in a1 deklariert

x = 6;
do {
  cout << feld[--x];
} while (feld[x] > 'b');
cout << endl;
```

Antwort: _____

a_3) *[1 Punkt]*

```
// char feld[] = "number"; ist in a1 deklariert

x = 6;
if (feld[1] == 'u')
  if (x < 5 && feld[3] == 'm')
    cout << "A";
  else
    if (x > 6 || feld[0] == 'r')
      cout << "B";
    else
      cout << "C";
else
  cout << "D";
return 0;
}
```

Antwort: _____

b) *[4 Punkte]* Geben Sie für das unten angegebene Programm an, welche Ausgabe erzeugt wird.

```
#include <iostream>
using namespace std;

int x = 3;

int funk1(int x) {
  x += 2;
  return x;
}
int funk2(int& x) {
  x *= 3;
  return x;
}
int main() {
  cout << x << "␣";
  cout << funk2(x) << "␣";
  cout << funk1(x) << "␣";
  cout << funk1(x) << endl;
  int x = 10;
  cout << x << "␣";
  cout << funk2(x) << "␣";
  cout << funk1(x) << "␣";
  cout << funk1(x) << endl;
  return 0;
}
```

Antwort:

Aufgabe 2: ***Grammatik und Automaten*** [10 Punkte]

Gegeben sei der endliche Automat $(S, \Sigma, \delta, F, s_0)$ mit der Zustandsmenge $S = \{0, 1, 2, 3, 4\}$ und dem Eingabealphabet $\Sigma = \{0, 1\}$, der Übergangsfunktion δ, die durch die folgende Tabelle gegeben ist, der Menge $F = \{0\}$ von Endzuständen und dem Anfangszustand $s_0 = 0$.

Hinweis: Der Automat erkennt genau die durch 5 teilbaren Binärzahlen (als Worte über den Eingabesymbolen 0 und 1).

$\Sigma \rightarrow$ $S \downarrow$	0	1
0	0	1
1	2	3
2	4	0
3	1	2
4	3	4

a) *[2 Punkte]* Geben Sie eine grafische Darstellung des endlichen Automaten an.

b) *[2 Punkte]* Geben Sie ein möglichst kurzes Wort (Folge der Eingabesymbole aus Σ) an, das von dem Automaten akzeptiert/erkannt wird und dabei alle Zustände durchläuft. Begründen Sie, warum es kein kürzeres Wort gibt.

c) *[3 Punkte]* Verändern Sie den Automaten so, dass nur noch Binärzahlen ohne führende Nullen akzeptiert werden. Worte wie 00 und 0101 sollen also von dem modifizierten Automaten nicht mehr akzeptiert werden, während 0 alleine akzeptiert wird. Geben Sie die erforderlichen Modifikationen an der Zustandsmenge, dem Startzustand und der Endzustandsmenge an, sowie die graphische Darstellung der „neuen“ Übergangsfunktion.

d) *[3 Punkte]* Geben Sie eine kontextfreie Grammatik G=(N,T,S,P) an, mit der genau die Binärzahlen ohne führende Nullen erzeugt werden können, die durch 4 teilbar sind. Zum Beispiel sollen also Worte wie 00 oder 0100 oder 101 nicht durch der Grammatik erzeugbar sein, während 0 alleine oder auch 100 erzeugt werden können.

Hinweis: Durch 4 teilbare Binärzahlen sind anhand der letzten beiden Binärziffern erkennbar.

Aufgabe 3: ***Zeiger und Rekursion*** [10 Punkte]

(a) *[5 Punkte]* Im folgenden Hauptprogramm werden (zugegebenermaßen etwas kompliziert) die Werte von Variablen mit Hilfe von Zeigern verändert.

```
int main() {
    // Die Adresse von a[0] ist 1000
    int a[] = { 40, 60, 100 };
    int b = 10;          // &b ist 2000
    int *p = &b;
    int *q = a;
    // Schritt 1

    *q = *p;
    // Schritt 2

    q = p;
    p++;
    p = &a[1];
    *p = 20;
    // Schritt 3

    p += 1;
    *q = *p;
    // Schritt 4

    *p = *(p+1)**q;
    // Schritt 5

    *(p+1) = *p + 1;
    ++(*(p-1));
    *q = *(p-1) + 2;
    // Schritt 6

    delete q; return 0;
}
```

Hinweis: Die Werte der Variablen nach Schritt 1 sind bereits vorgegeben. Dadurch sind insbesondere auch die *Adressen* der Variablen gegeben. (Ein `int` hat 4 Bytes.)

Wir interessieren uns für die Werte der Variablen nach der Ausführung der mit `Schritt 1`, `Schritt 2` usw. gekennzeichneten Zeilen. Vervollständigen Sie die folgende Tabelle:

Schritt	a[0]	a[1]	a[2]	p	*p	q	*q
1	40	60	100	2000	10	1000	40
2							
3							
4							
5							
6							

b) *[4 Punkte]* Gegeben sei die folgende Funktion $p : \mathbb{N}_0 \times \mathbb{N}_0 \to \mathbb{N}_0$:

$$p(n,k) = \begin{cases} 0 & \text{falls } n = 0 \text{ oder } k > n \\ 1 & \text{falls } k = 0 \text{ oder } k = n \\ p(n-1,k) + p(n-1,k-1) & \text{sonst} \end{cases}$$

Anmerkung: Die Funktion beschreibt die Anzahl der Möglichkeiten k Elemente ohne Berücksichtigung der Reihenfolge aus einer n-elementigen Menge auszuwählen.

b$_1$) *[1 Punkt]* Schreiben Sie zuerst den Kopf (Prototypen) der Funktion auf und legen Sie geeignete Rückgabe- sowie Parametertypen fest.

b$_2$) *[3 Punkte]* Implementieren Sie die Funktion p:

c) *[1 Punkt]* Was berechnet folgende rekursive Funktion?

```
unsigned long fa(unsigned long n) {
  return (n == 0) ? 1 : fa(n - 1) * n;
}
```