Aufgabenblatt zur 6. Übung

Zeitraum: 23.11. bis 27.11.2009

1. Aufgabe: Klausuraufgabe 07.2009 (AGS 4.17)

Gegeben sei folgendes *C*-Programm:

```
 1  #include <stdio.h>
 2  void h (int *x);
 3
 4  void g (int a, int *b, int *c)
 5  { /*label1*/
 6    if (a == 1)
 7      h(c);  /*$1*/
 8    else
 9      *c = *b + 2;
10    /*label2*/
11  }
12
13  void h (int *x)
14  { int y;
15    y = 3;
16      /*label3*/
17    g(y, &y, x);  /*$2*/
18      /*label4*/
19  }
20
21  int main ()
22  {
23    int e, a;
24    scanf("%i", &e);
25      /*label5*/
26    g(e, &e, &a);  /*$3*/
27      /*label6*/
28    printf("%d", a);
29    return 0;
30  }
```

(a) Geben Sie den Gültigkeitsbereich jedes Objektes des Programms an.

(b) Stellen Sie eine Rechnung des Programms für die Eingabe e = 1 als pulsierenden Speicher dar, wobei die aktuelle Situation bei jedem Passieren der Marken `label1` bis `label6` gezeigt werden soll. Dokumentieren Sie hierzu jeweils alle sichtbaren Variablen mit ihren Wertebelegungen. Hat eine Variable zu einem Zeitpunkt noch keinen Wert erhalten, so geben Sie anstelle des Wertes ein ? an (also z. B. x=?, wenn x zum Zeitpunkt der Protokollierung noch keinen Wert besitzt). Führen Sie des Weiteren auch ein Rücksprungmarkenprotokoll.

Beachten Sie: `$1$` bis `$3$` seien die bereits festgelegten Rücksprungmarken.

2. Aufgabe: (AGS 3.9)

Es soll geprüft werden, ob der Inhalt des Zeichenfeldes `feld` der Länge `l` ein Palindrom ist, das heißt, ob die Zeichenkette sowohl von vorne als auch von hinten gelesen gleich lautet.
Ist die Zeichenkette ein Palindrom, so soll der Parameter `korrekt` den Wert `true` repräsentieren, sonst `false`.

(a) Von folgender Funktion wird behauptet, dass sie diese Aufgabe löst:

```
Palindrom1(char feld[], int l, int korrekt) {
  int i;
  i = 1; l = l - 1;
  while (i < l && korrekt) {
    korrekt = feld[i] == feld[l];
    i = i + 1;
  }
  return korrekt;
}
```

Prüfen Sie diese Funktion. Korrigieren Sie eventuell enthaltene Fehler.

(b) Schreiben Sie eine Funktion `Palindrom2`, die rekursiv arbeitet. Überlegen Sie sich hierbei als erstes einen geeigneten Funktionskopf.

3. Aufgabe: (AGS 4.13)

Gegeben sei das folgende *C*-Programm:

```
(1)  #include <stdio.h>
(2)
(3)  int e;
(4)
(5)  void g(int y, int z, int *x);
(6)
(7)  void f(int y, int *z) {
(8)    int u; /* label1 */
(9)    if (y > 0) {
(10)     f(y - 1, &u); /* $2 */ /* label2 */
(11)     g(y - 1, u, z); /* $3 */
(12)   }
(13)   else *z = 2;
(14)   /* label3 */
(15) }
(16)
(17) void g(int y, int z, int *x) {
(18)   int u; /* label4 */
(19)   if (y > 0) {
(20)     f(y - 1, &u); /* $4 */ /* label5 */
(21)     *x = u + z;
(22)   }
(23)   else *x = 5;
(24)   /* label6 */
(25) }
(26)
(27) int main() {
(28)    int a;
(29)    scanf("%i", &e); /* label7 */
(30)    f(e, &a); /* $1 */ /* label8 */
```

```
(31)    printf("a = %d\n", a);
(32)    return 0;
(33) }
```

(a) Geben Sie den Gültigkeitsbereich jedes Objektes des Programmes an. Nutzen Sie dazu die angegebenen Zeilennummern.

(b) Stellen Sie die Rechnung des Programmes für die Eingabe `e = 1` als pulsierenden Speicher dar, wobei die aktuelle Situation bei jedem Passieren der Marken `label1` bis `label8` dokumentiert werden soll. Dokumentieren Sie hierzu jeweils alle sichtbaren Variablen mit ihren Wertebelegungen. Führen Sie des Weiteren auch ein Rücksprungmarkenprotokoll.
Hat eine Variable noch keinen Wert erhalten, so geben Sie anstelle des Wertes ein ? an (also z. B. x = ?, wenn x zum Zeitpunkt der aktuellen Marke noch unbelegt ist). Beachten Sie weiter: `$1` bis `$4` sind die bereits festgelegten Rücksprungmarken.

Zusatzaufgabe: (AGS 4.2*)

Gegeben sei das folgende *C*–Programm:

```
1  #include <stdio.h>
2  int a, b, c;
3
4  void g(int x, int y, int *z) {
5    /*label1*/
6    if (x > 0) {
7      g(x - 1, y, z); /*$2*/ /*label7*/
8      *z = *z + 1;
9    }
10   else *z = y;
11 }
12
13 void f(int x, int y, int *z) {
14   int u; /*label2*/
15   if (x > 0) {
16     f(x - 1, y, &u); /*$3*/ /*label3*/
17     g(u, y, z); /*$4*/ /*label4*/
18   }
19   else *z = 0;
20 }
21
22 int main() {
23   printf("\n Zahl a: ");
24   scanf("%d", &a);
25   printf("\n Zahl b: ");
26   scanf("%d", &b); /*label5*/
27   f(a, b, &c); /*$1*/ /*label6*/
28   printf("\nDas Ergebnis lautet: %d\n\n", c);
29   return 0;
30 }
```

(a) Geben Sie den Gültigkeitsbereich jedes Objektes des Programmes an.

(b) Stellen Sie eine Rechnung des Programms für die Eingaben `a = 2` und `b = 2` als pulsierenden Speicher dar, wobei die aktuelle Situation bei jedem Passieren der Marken `label1` bis `label7` gezeigt werden soll. Dokumentieren Sie hierzu jeweils alle sichtbaren Variablen mit ihren Wertebelegungen. Hat eine Variable zu einem Zeitpunkt noch keinen Wert erhalten, so geben Sie anstelle des Wertes ein ? an (also z. B. x=?, wenn x zum Zeitpunkt der Protokollierung noch keinen Wert besitzt). Führen Sie des Weiteren auch ein Rücksprungmarkenprotokoll.

Beachten Sie: `$1` bis `$4` sind die bereits festgelegten Rücksprungmarken.