Aufgabenblatt zur 6. Übung

Zeitraum: 21.11. bis 25.11.2011

1. Aufgabe: (AGS 4.15)

Gegeben sei folgendes *C*-Programm:

```
 1  #include <stdio.h>}
 2
 3  void h(int *x, int y) {
 4      /* label1 */
 5      if (*x > y)
 6          h(x, *x); /* $1 */
 7      else
 8          *x = *x - y;
 9      /* label2 */
10  }
11
12  void g(int *a) {
13      int b;
14      /* label3 */
15      b = *a + 1;
16      while (b >= 0) {
17          h(&b, *a); /* $2 */
18          *a = b + 1;
19          /* label4 */
20      }
21  }
22
23  int main() {
24      int z;
25      scanf("%i", &z);
26      /* label5 */
27      g(&z); /* $3 */
28      /* label6 */
29      printf("%d", z);
30      return 0;
31  }
```

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

(b) Setzen Sie das Speicherbelegungs- und Rücksprungmarkenprotokoll der untenstehende Tabelle fort. Dokumentieren Sie die aktuelle Situation beim Passieren der Marken `label1` bis `label6`. Geben Sie jeweils den Rücksprungmarkenkeller und die *sichtbaren* Variablen mit ihrer Wertebelegung an. Die Inhalte von Speicherzellen nicht-sichtbarer Variablen brauchen Sie nur bei Änderungen einzutragen. Beachten Sie: `$1` bis `$3` seien die bereits festgelegten Rücksprungmarken.

Label	Rücksprungmarken	1	2	3	4	5	6	7	8	9	10
label5	–	z 2									
label3	3		a #1	b ?							

2. Aufgabe: (AGS 3.14)

Definieren Sie einen Datentyp `IntList`, der beliebig lange einfach verkettete Listen ganzer Zahlen darstellen kann. Implementieren Sie eine Funktion `void Insert(IntList *l, int n)`, die in eine bereits sortierte Liste `l` eine ganze Zahl `n` einsortiert. Schreiben Sie anschließend ein kleines Hauptprogramm zum Test der Funktion `Insert`.

3. Aufgabe: (AGS 3.15)

Implementieren Sie entsprechend der 2. Aufgabe eine Funktion `void Insert2(IntList *l, int n)`, die nach einer rekursiven Lösungsstrategie arbeitet. Vergleichen Sie anschließend die rekursive mit der iterativen Lösung.

Zusatzaufgabe: (AGS 4.9*)

Gegeben sei das folgende *C*-Programm:

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

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

(b) Stellen Sie eine Rechnung des Programms für die Eingabe `e` = 2 als pulsierenden Speicher dar, wobei die aktuelle Situation bei jedem Passieren der Marken `label1` bis `label6` gezeigt werden soll. Dokumentieren Sie jeweils alle *sichtbaren* Variablen mit ihrer Wertebelegung. Falls die Variable zu diesem Zeitpunkt noch keinen Wert erhalten hat, so tragen Sie anstelle des Wertes ein „?“ ein. Die Inhalte von Speicherzellen nicht-sichtbarer Variablen brauchen Sie nur bei Änderungen einzutragen. Nutzen Sie für das Speicherbelegungs- und Rücksprungmarkenprotokoll die untenstehende Tabelle (die ersten beiden Zeilen sind bereits vollständig ausgefüllt). Beachten Sie: `$1` bis `$3` seien die bereits festgelegten Rücksprungmarken.