『プログラム言語論』 期末試験
2013 年 6 月 27 日 (木)

問 1. (配点 25 点) MiniC 言語で書かれた次のプログラムについて以下の問に答えよ.

```
int x;
int f (int y) {
  return x + y + y;
}
int g (int z) {
  int x; int y;
  x = 20; y = f(0);
  print y;            /* C 言語の printf("%d\n", y); と同じ*/
  return 30;
}
int main () {
  x = 10;
  print f(g(0));
  print g(g(0));
}
```

上記のプログラムを以下のそれぞれの方式で実行すると、どのような値が印刷されるかを示しなさい.

1-a. 静的束縛かつ値呼び : **答.** 10,70,10,10,30

1-b. 静的束縛かつ名前呼び : **答.** 10,10,70,10,30 (注。以前に、この場所に載せていた解答例では 10,10,70,10,10,30 と書いていましたが、正しくは、上記のものです。こちらは単純な誤記でした。指摘してくれた小嶋君に感謝します。)

1-c. 静的束縛かつ必要呼び : **答.** 10,70,10,30

1-d. 動的束縛かつ値呼び : **答.** 20,70,20,20,30

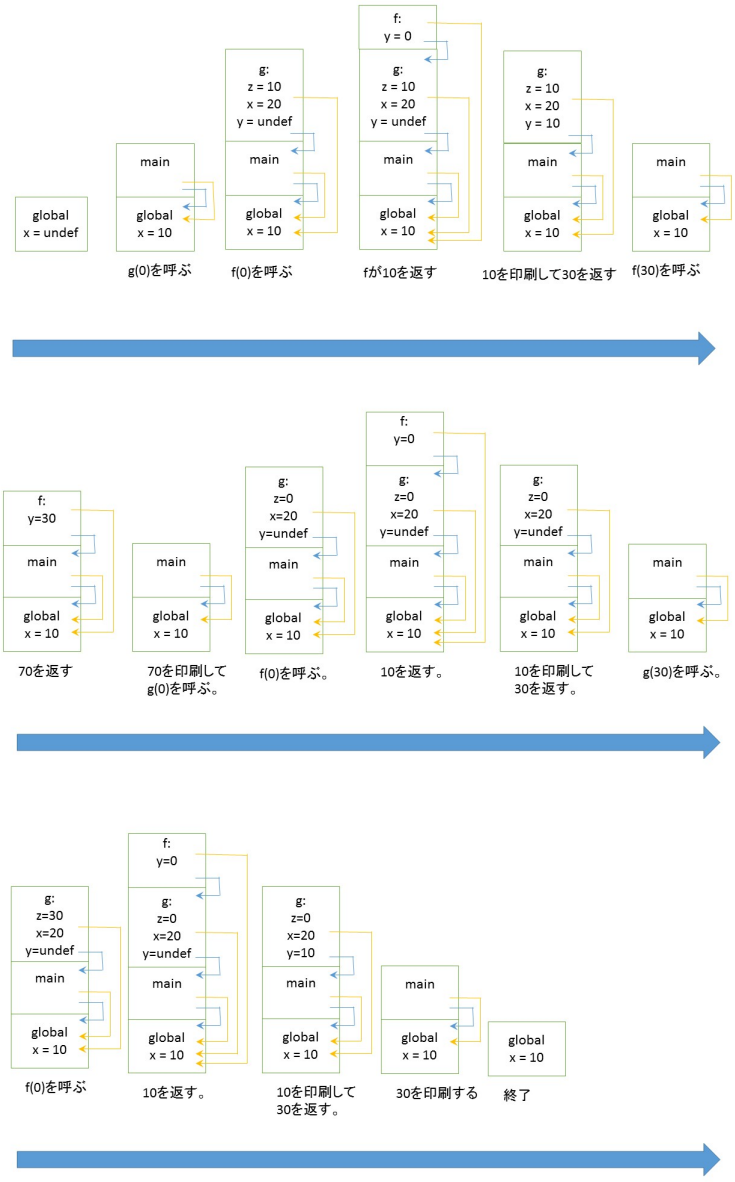
(注。以前に、この場所に載せていた解答例では 20,80,20,20,30 と書いていましたが、正しくは、上記のものです。こちらは単純な誤記ではなく、私の間違いです。指摘してくれた、湛君に感謝します。)

1-e. 上記のプログラムを静的束縛かつ値呼び方式で実行した時のスタックの状態の変化を図示しなさい。

なお，1 回の関数呼び出しごとに、1 つのスタックフレームがスタックに積まれるとし、また、スタックフレームの中身としては、局所変数とその値、Access Link, Control Link の 3 つのデータのみを書けばよい。

答 (略解).

実行の各時点におけるスタックの状態を、左から右に表示する。



問 2. **(配点 25 点)** 次の MiniML プログラムについて考える.

```
let f x = x + 7 in
let rec g n =
  if n = 1 then f 10
  else
    let y = g (n + (-1)) in
       f y
in
let rec h n =
  if n = 1 then f
  else
    let z = h (n + (-1)) in
       fun x -> f (z x)
in
let a1 = f 3 in
let a2 = g 5 in
let a3 = h 10 in
let a4 = a3 9 in
 (print a2, print a4) ;; (* a2,a4 の値を印刷 *)
```

このとき，以下の問に答えなさい．

2-a. このプログラムを MiniML で静的束縛かつ値呼び方式で実行する時，最後に印刷される a2 と a4 の値を示しなさい。 (a1, a3 の値は不要。)

答. a2=45 a4=79

2-b. 関数 h が返す値 (たとえば a3 に代入される値) は関数である。このような関数 h を何というか，また、そのような関数を定義できることの利点 (merit) を 1-2 行で簡潔に述べなさい。

答. 高階関数。 関数を引数に取ったり、関数を返り値とする関数を高階関数と言い、map 関数などが代表例である。高階関数を適切に使うことにより、同一の処理を 1 回書けば済むようになる (プログラムの抽象化を高める)、プログラムをより構造化したものにできるなどの利点がある。

2-c. 前問 (2-b) のような関数を処理する方式として，関数クロージャがある．関数クロージャは，通常，ヒープ上に作成されるが，もし，これがスタックに作成されたとしたら，どのような問題が発生するか簡潔に説明せよ。

答. スタック上のデータは、そのデータを含むスタックフレームに対応する関数呼び出しが終了すると、なくなってしまうので、関数を返り値に含む場合等において、適切な処理ができなくなる。

2-d. 上記のプログラムにおける関数 `g` と関数 `h` の型を述べよ．

答. g の型は `int -> int` であり、h の型は `int -> (int -> int)` である。

問 3. (配点 20 点) 次の抽象構文で定まる式からなる言語 L を考える．ただし，n は -10 や 3 などの整数定数を表す．

$$e ::= \texttt{Int}(n) \mid \texttt{Plus}(e, e) \mid \texttt{Minus}(e, e)$$

L に対して，以下の動作 (遷移規則) をする抽象機械 (CK 機械) を考える．ただし，初期状態は，$\langle eval,\ e,\ \texttt{init}\rangle$ である．また，一部を省略して $\cdots$ と記載している．

$$
\begin{aligned}
\langle eval,\ \texttt{Int}(n),\ K\rangle &\to \langle apply,\ K,\ n\rangle \\
\langle eval,\ \texttt{Plus}(e_1, e_2),\ K\rangle &\to \langle eval,\ e_1,\ (\texttt{::}(plus1, e_2), K)\rangle \\
\langle eval,\ \texttt{Minus}(e_1, e_2),\ K\rangle &\to \cdots \\
\langle apply\ , (\texttt{::}(plus1, e), K),\ n\rangle &\to \langle eval,\ e,\ (\texttt{::}(plus2, n), K)\rangle \\
\langle apply\ , (\texttt{::}(plus2, m), K),\ n\rangle &\to \langle apply,\ K,\ p\rangle \quad (\text{ただし } p = n \dot{+} m) \\
\langle apply\ , (\texttt{::}(minus1, e), K),\ n\rangle &\to \langle eval,\ e,\ (\texttt{::}(minus2, n), K)\rangle \\
\langle apply\ , (\texttt{::}(minus2, m), K),\ n\rangle &\to \langle apply,\ K,\ p\rangle \quad (\text{ただし } p = \cdots) \\
\langle apply\ , \texttt{init},\ n\rangle &\to n \quad (\text{最終結果})
\end{aligned}
$$

ここで $\dot{+}$ は，整数定数に対する加算 (addition) を表す．また、以下の問題の解答で，整数定数に対する減算 (引き算,subtraction) を表す $\dot{-}$ を使ってよい。さらに、K や `init` は (制御に関する情報をもつ) スタックを表す。$(\texttt{::}M, K)$ はスタック K に要素 M をプッシュして得られるスタックをあらわす。

このとき，以下の問に答えなさい．

3-a. 以下の初期状態から始まり，最終結果を得るまでの抽象機械の状態遷移を書きなさい．

$$\langle eval,\ \texttt{Plus}(\texttt{Int}(10), \texttt{Int}(20)),\ \texttt{init}\rangle$$

答. 以下の通り。

$$
\begin{aligned}
\langle eval,\ \texttt{Plus}(\texttt{Int}(10), \texttt{Int}(20)),\ \texttt{init}\rangle &\to \langle eval,\ \texttt{Int}(10),\ (\texttt{::}(plus1, \texttt{Int}(20)),\ \texttt{init})\rangle \\
&\to \langle apply,\ (\texttt{::}(plus1, \texttt{Int}(20)),\ \texttt{init}),\ 10\rangle \\
&\to \langle eval,\ \texttt{Int}(20),\ (\texttt{::}(plus2, 10), \texttt{init})\rangle \\
&\to \langle apply,\ (\texttt{::}(plus2, 10), \texttt{init}),\ 20\rangle \\
&\to \langle apply,\ \texttt{init},\ 30\rangle \\
&\to 30
\end{aligned}
$$

3-b. 上記の記述では，$\texttt{Minus}(e_1, e_2)$ に対する動作の一部が「$\cdots$」となっている。式 $\texttt{Minus}(e_1, e_2)$ の評価は、(1) e_1 を評価し，(2) e_2 を評価し、(3) 最終的に e_1 の評価結果から e_2 の評価を引いた数を返す、とする時，2 か所の「$\cdots$」を適切に埋めなさい．

答. 以下の通り。

$$
\begin{aligned}
\langle eval,\ \texttt{Minus}(e_1, e_2),\ K\rangle &\to \langle eval,\ e_1,\ (\texttt{::}(minus1, e_2), K)\rangle \\
\langle apply\ , (\texttt{::}(minus2, m), K),\ n\rangle &\to \langle apply,\ K,\ p\rangle \quad (\text{ただし } p = m \dot{-} n)
\end{aligned}
$$

問 4. (配点 30 点) 以下の事項から 3 つを選び、それぞれ 3 行程度で説明しなさい。

4-a. インタープリタとコンパイラの違い (入出力の違いだけでなく，通常，性能が大きく違うことについてその理由を簡潔に述べよ．)

略解. インタープリタは解釈系あるいは評価系とも言われ、プログラムとその入力となるデータを受け取り、(もし停止すれば) 計算結果を返すプログラムである。通常、プログラムの静的情報の収集等を行わないので、実行速度は遅い。一方、コンパイラは翻訳系とも言われ、プログラムを受け取り、(必ず停止して) プログラムを返すプログラムである。コンパイラが返すプログラムは、(入力となる) データを受け取り、(もし停止すれば) 計算結果を返す。多くの場合、プログラムの静的情報の収集等を行い、それにより最適化を施したプログラムを生成するため、コンパイルされたプログラムの実行はインタープリタの実行より高速である。

4-b. オブジェクト指向言語における継承とサブタイピングについて．(それらが何を表し，どう違うか．)

略解. (省略; 授業スライドを参照すること)

4-c. 多相型 (polymorphic type) について (それが何か，どのような種類があるか，どのような利点があるか．)

略解. (省略; 授業スライドの「補足」の部分を参照すること)

4-d. 静的型付けと動的ルックアップの両立 (Java では，動的ルックアップであるのに，型付けは静的に行なっている．なぜ，それが可能か．)

略解. (前置き)Java はオブジェクト指向言語であるため、メソッド名からメソッド本体を探す「ルックアップ」が、動的に行われる。よって、どのメソッドに起動されるかは、実行時にならないとわからない (静的にわかることもあるが、すべてが静的にわかるわけではない)。(前置き終わり)

コンパイル時に (静的に) メソッド実体がわからなくても型付けがおこなえるようにするため、Java では、「子クラスにおいてメソッドの実装を上書きする (override する) ときは、その引数の個数、引数の型、返り値の型は、すべて親クラスのメソッドと同一でなければいけない」という制約をつけている。これにより、型に関する限り、親クラスのメソッドが呼ばれようと、子クラスのメソッドが呼ばれようと、同一の結果となるため、支障はきたさない。

4-e. 抽象データ型と情報の隠蔽について (それらは何か，どういう利点があるか．)

略解. (省略; 授業スライドを参照すること)

以上.