论题 1-5 作业

姓名：陈劭源　　学号：161240004

1 [DH] Problem 2.10

Let T be a vector of Booleans.

(1) for I going from 1 to N do the following:
  (1.1) $T[I] \leftarrow$ false;
(2) for I going from 1 to N do the following:
  (2.1) if $P[I] < 1$ or $P[I] > N$ do the following:
    (2.2.1) output 'NO';
    (2.2.2) end;
  (2.2) $T[P[I]] =$ true;
(3) for I going from 1 to N do the following:
  (3.1) if $T[I] =$ false do the following:
    (3.1.1) output 'NO';
    (3.1.2) end;
(4) output 'YES'.

2 [DH] Problem 2.11

Let K be a vector of Booleans, L be a vector of integers which stores a permutation.

subroutine **produce permutation** I
(1) if $I = N$ do the following:
  (1.1) output R;
  (1.2) return;
(2) for i going from 1 to N do the following:
  (2.1) if $K[i]$ is false do the following:
    (2.1.1) $R[I] \leftarrow i$
    (2.1.2) $K[i] \leftarrow$ true;
    (2.1.3) call **produce permutation** $I+1$
    (2.1.4) $K[i] \leftarrow$ false.

(1) i going from 1 to N do the following:

(1.1) $K[i] \leftarrow$ false;

(2) call **produce permutation** 0.

3 [DH] Problem 2.12

(a) i. **read**(X), **push**(X,S), **read**(X), **push**(X,S), **read**(X), **print**(X), **pop**(X,S), **print**(X), **pop**(X,S), **print**(X)

ii. **read**(X), **push**(X,S), **read**(X), **push**(X,S), **read**(X), **print**(X), **read**(X), **print**(X), **pop**(X,S), **print**(X), **pop**(X,S), **print**(X)

iii. **read**(X), **push**(X,S), **read**(X), **push**(X,S), **read**(X), **print**(X), **read**(X), **push**(X,S), **read**(X), **print**(X), **read**(X), **push**(X,S), **read**(X), **print**(X), **pop**(X,S), **print**(X), **read**(X), **print**(X), **pop**(X,S), **print**(X), **read**(X), **print**(X), **pop**(X,S), **print**(X)

(b) i. 要生成 (3, 1, 2) 这个排列，由于 3 是最先输出的，1, 2 依次在栈中，此时若要继续输出，必然是以 2,1 的形式输出，所以不可能用栈生成 (3, 1, 2) 这个排列。 □

ii. 要生成 (4, 5, 3, 7, 2, 1, 6) 这个排列，当输出 7 时，栈中剩余的元素依次为 1, 2, 6，下一个需要输出 2，但输出 2 之前 6 必须输出，从而不可能用栈生成 (4, 5, 3, 7, 2, 1, 6) 这个排列。 □

(c) 容易验证，以下排列可以用栈生成：

(1, 2, 3, 4) (1, 2, 4, 3) (1, 3, 2, 4) (1, 3, 4, 2) (1, 4, 3, 2) (2, 1, 3, 4) (2, 1, 4, 3)
(2, 3, 1, 4) (2, 3, 4, 1) (2, 4, 3, 1) (3, 2, 1, 4) (3, 2, 4, 1) (3, 4, 2, 1) (4, 3, 2, 1)

以下排列不能用栈生成：

(1, 4, 2, 3) (2, 4, 1, 3) (3, 1, 2, 4) (3, 1, 4, 2) (3, 4, 1, 2) (4, 1, 2, 3) (4, 1, 3, 2)
(4, 2, 3, 1) (4, 2, 1, 3) (4, 3, 1, 2)

所以共有 10 个排列不能用栈生成。

4 [DH] Problem 2.13

Let S be an empty stack. Assume that the length of the permutation is N.

function **test**

(1) $I \leftarrow 1$;

(2) for i going from 1 to N do the following:

(2.1) **read**(X);

(2.2) if $I \leq X$ do the following:

(2.2.1) for j going from I to X do the following:

(2.2.1.1) **push**(j,S);

(2.2.2) **pop**(X,S);

(2.2.3) $I \leftarrow X+1$;

(2.3) otherwise do the following:

(2.3.1) **pop**(Y,S);

(2.3.2) if $X \neq Y$ then return false;

(3) return true.

subroutine **print operations**

(1) $I \leftarrow 1$;

(2) for i going from 1 to N do the following:

(2.1) **read**(X);

(2.2) if $I \leq X$ do the following:

(2.2.1) for j going from I to X do the following:

(2.2.1.1) **print**("**read**(X)");

(2.2.1.2) **print**("**push**(X,S)");

(2.2.1.3) **push**(j,S);

(2.2.2) **print**("**pop**(X,S)");

(2.2.3) **print**("**print**(X)");

(2.2.4) **pop**(X,S);

(2.2.5) $I \leftarrow X+1$;

(2.3) otherwise do the following:

(2.3.1) **pop**(Y,S);

(2.3.2) **print**("**pop**(X,S)");

(2.3.3) **print**("**print**(X)").

(1) if **test** is true then do the following:

(1.1) **print**("Yes");

(1.2) call **print operations**;

(2) otherwise do the following:

(2.1) **print**("No").

5 [DH] Problem 2.14

(a) i. Obtain by a queue: **read**(X), **add**(X,Q), **read**(X), **add**(X,Q), **read**(X), **print**(X), **remove**(X,Q), **print**(X), **remove**(X,Q), **print**(X)

Obtain by two stacks: **read**(X), **push**(X,S), **read**(X), **push**(X,S'), **read**(X),

print(X), **pop**(X,S), **print**(X), **pop**(X,S'), **print**(X)

ii. Obtain by a queue: **read**(X), **add**(X,Q), **read**(X), **add**(X,Q), **read**(X), **add**(X,Q), **read**(X), **print**(X), **read**(X), **print**(X), **remove**(X,Q), **add**(X,Q), **remove**(X,Q), **add**(X,Q), **remove**(X,Q), **print**(X), **read**(X), **add**(X,Q), **read**(X), **print**(X), **remove**(X,Q), **add**(X,Q), **remove**(X,Q), **print**(X), **remove**(X,Q), **add**(X,Q), **remove**(X,Q), **print**(X), **remove**(X,Q), **print**(X)

Obtain by two stacks: **read**(X), **push**(X,S), **read**(X), **push**(X,S), **read**(X), **push**(X,S), **read**(X), **print**(X), **read**(X), **print**(X), **pop**(X,S), **print**(X), **read**(X), **push**(X,S'), **read**(X), **print**(X), **pop**(X,S), **print**(X), **pop**(X,S), **print**(X), **pop**(X,S'), **print**(X)

(b) 用以下方法可只用一个队列生成任一排列：对于某一个要输出的数字，如果不在队列中，则将该数字之前的数字全部入队，然后直接输出该数字；如果在队列中，反复将队首数字出队后再入队“翻找”整个队列，直到找到为止，输出这个数字。对于排列中的每一个数字，依次重复上述操作，即可用一个队列输出任意排列。 □

(c) 用以下方法可只用两个队列生成任一排列：对于某一个要输出的数字，如果不在任一栈中，则将该数字之前的数字全部压入任何一个栈，然后直接输出该数字；如果在某个栈中，将该数字之上的数字依次弹出并压入到另一个之中，然后输出这个数字。对于排列中的每一个数字，依次重复上述操作，即可用两个栈输出任意排列。 □

6 [DH] Problem 2.15

Let S, S' be two empty stacks. Assume that the length of the permutation is N.

function **top**(S)
(1) **pop**(t,S);
(2) **push**(t,S);
(3) return t.

(1) $I \leftarrow 1$;
(2) for i going from 1 to N do the following:
  (2.1) **read**(X);
  (2.2) if $I \leq X$ do the following:
    (2.2.1) for j going from I to X do the following:
      (2.2.1.1) **print**(”**read**(X)”);
      (2.2.1.2) **print**(”**push**(X,S)”);
      (2.2.1.3) **push**(j,S);
    (2.2.2) **print**(”**pop**(X,S)”);
    (2.2.3) **print**(”**print**(X)”);

(2.2.4) **pop**(X, S);

(2.2.5) $I \leftarrow X + 1$;

(2.3) otherwise do the following:

(2.3.1) while **is-empty**(S') is false do the following:

(2.3.1.1) **print**("**pop**(X, S')");

(2.3.1.2) **print**("**push**(X, S)");

(2.3.1.3) **pop**(X, S');

(2.3.1.4) **push**(X, S);

(2.3.2) while **top**(S) $\neq I$ do the following:

(2.3.2.1) **print**("**pop**(X, S)");

(2.3.2.2) **print**("**push**(X, S')");

(2.3.2.3) **pop**(X, S);

(2.3.2.4) **push**(X, S');

(2.3.3) **pop**(X, S);

(2.3.4) **print**("**pop**(X, S)");

(2.3.5) **print**("**print**(X)").

7 [DH] Problem 2.16

(a) Let T be an empty binary search tree, L be a list of integers, N be the number of integers in L.

subroutine **add** X **to** S

(1) if S is empty then do the following:

(1.1) $S \leftarrow X$;

(1.2) return;

(2) if $X < S$ then do the following:

(2.1) **add** X **to left**(S);

(3) otherwise do the following:

(3.1) **add** X **to right**(S).

(1) for i going from 1 to N do the following:

(1.1) **add** $L[i]$ **to** T.

(b) Let T be a binary search tree.

subroutine **visit** S

(1) if S is empty then return;

(2) **visit right**(S);

(3) output S;
(4) **visit left**(S).

(1) **visit left**(T).