— *Solution notes* —

COMPUTER SCIENCE TRIPOS Part IA – 2012 – Paper 1

1 Foundations of Computer Science (LCP)

lecture 8, general programming skills

Recall that a dictionary of (*key*, *value*) pairs can be represented by a binary search tree. Define the *union* of two binary search trees to be any binary search tree consisting of every node of the given trees.

(*a*) Write an ML function `union` to return the union of two given binary search trees. [*Note:* You may assume that they have no keys in common.] [6 marks]

Answer: This particular solution uses the general update function, but there could be many alternative solutions. Taking this function for granted (in other words, omitting it from the solution) isn't acceptable, as that would make the problem almost trivial.

```
fun update (Lf, b: string, y) = Br((b,y), Lf, Lf)
  | update (Br((a,x),t1,t2), b, y) =
      if b<a then  Br ((a,x),  update(t1,b,y),  t2)
      else
      if a<b then  Br ((a,x),  t1,  update(t2,b,y))
      else (*a=b*) Br ((a,y),t1,t2);

fun union (Lf, t) = t
  | union (Br((a,z), t1, t2), t) = union (t1, union(t2, update(t,a,z)));
```

Define a *slice* of a binary search tree to be a binary search tree containing every (*key*, *value*) node from the original tree such that $x \leq key \leq y$, where x and y are the given endpoints.

(*b*) Write an ML function `takeSlice` to return a slice – specified by a given pair of endpoints – from a binary search tree. [4 marks]

Answer: The solution is a straightforward recursion.

```
fun takeSlice (x,y: string) Lf = Lf
  | takeSlice (x,y) (Br((a,z),t1,t2)) =
      if y<a then takeSlice (x,y) t1
      else if a<x then takeSlice (x,y) t2
      else Br((a,z), takeSlice (x,y) t1, takeSlice (x,y) t2);
```

(*c*) Write an ML function `dropSlice` to *remove* a slice from a binary search tree: given a tree and a pair of endpoints, it should return the binary search tree consisting of precisely the nodes such that $x > key$ or $key > y$. [*Hint:* First consider the simpler task of deleting a node from a binary search tree.] [8 marks]

Answer: Deletion is not straightforward. The problem is to combine the remaining subtrees while preserving the ordering. A simple approach is to attach the right-hand tree at the far-right end of the left-hand tree, but inevitably, the resulting tree will be unbalanced.

Given deletion, the solution is once again a straightforward recursion.

```
fun join (Lf, t) = t
```

```
  | join (Br(a,t1,t2), t) = Br (a, t1, join(t2,t));

fun dropSlice (x,y: string) Lf = Lf
  | dropSlice (x,y) (Br((a,z),t1,t2)) =
      if y<a then Br((a,z), dropSlice (x,y) t1, t2)
      else if a<x then Br((a,z), t1, dropSlice (x,y) t2)
      else join (dropSlice (x,y) t1, dropSlice (x,y) t2);
```

(*d*) The tree t need not be identical to that returned by

```
union(takeSlice(t, x, y),
      dropSlice(t, x, y))
```

Briefly explain how such an outcome is possible. [2 marks]

Answer: They will represent equivalent dictionaries, in that they map the same values to the same keys. However, many distinct binary search trees can represent any particular dictionary. It's highly unlikely that the operation described in the question would preserve the exact structure of a binary search tree.

[*Note:* All ML code must be explained clearly and should be free of needless complexity.]