The University of Nottingham

SCHOOL OF COMPUTER SCIENCE

A LEVEL 4 MODULE, AUTUMN 2011–2012

ALGORITHM DESIGN

Time allowed 2 hours

Candidates must NOT start writing their answers until told to do so.

Answer THREE questions.
Marks available for sections of questions are shown in brackets in the right-hand margin.

No calculators are permitted in this examination.

Dictionaries are not allowed with one exception. Those whose first language is not English may use a standard translation dictionary to translate between that language and English provided that neither language is the subject of this examination. Subject specification translation dictionaries are not permitted.

No electronic devices capable of storing and retrieving text, including electronic dictionaries, may be used.

DO NOT turn examination paper over until instructed to do so

1 Consider the following program.

```
{  $M<N \wedge \mathsf{f}(M)\le 0\le \mathsf{f}(N)$  }
$i,j := M,N$
{ **Invariant:**  $M\le i<j\le N \wedge \mathsf{f}(i)\le 0\le \mathsf{f}(j)$
  **Measure of progress:**  $j-i$ } ;
do $i+1\ne j$ $\rightarrow$  {  $i+1\ne j \wedge i<j \wedge \mathsf{f}(i)\le 0\le \mathsf{f}(j)$  }
                $k := (i+j)\div 2$
                {  $i<k<j \wedge \mathsf{f}(i)\le 0\le \mathsf{f}(j)$  } ;
                if  $\mathsf{f}(k)\le 0 \rightarrow i := k$
                $\square$  $0\le \mathsf{f}(k) \rightarrow j := k$
                fi
                {  $i<j \wedge \mathsf{f}(i)\le 0\le \mathsf{f}(j)$  }
od
{  $M\le i<N \wedge \mathsf{f}(i)\le 0\le \mathsf{f}(i+1)$  }
```

Construct a verification condition for each of the four assignment statements in the program and a verification condition for the test for termination of the loop. (5 marks will be awarded for each.) You do NOT have to verify the validity of the verification conditions. (25)

2 Design a program that performs an *in-situ* reversal of the elements of an array. (That is, the program may use only a constant amount of additional storage space; it is not allowed to copy the array into another array.) The program should be designed to work for arrays of arbitrary length (including zero, of course). You should assume that the procedure $swap$ is such that $swap(i,j)$ swaps the array values indexed by i and j, provided that i and j are within the bounds of the array. When i equals j, you may assume that the procedure call is equivalent to skip.

Assume as precondition of your program the property:

$$0\le N \wedge \langle \forall j : 0\le j<N : a[j]=a_0[j]\rangle$$

That is, assume that the input array is a and a_0 is a ghost variable used to relate the array a at any stage in the algorithm to its initial value.

Your program should be annotated with sufficient assertions that it is possible for an independent reader to determine its specification and formally verify that the specification is met by the program. In particular, the assertions should guarantee that no array-bound errors can occur. (10)

Augment your program so as to count the number of times that two unequal array values are swapped. Annotate your augmented program with assertions that express clearly the function of any new variable(s) that you introduce. (The annotation is the most important part of your solution; a mark of 0 will be awarded if no assertions are added.) (15)

3 Given are two integers M and N satisfying $0 \leq M$ and $0 \leq N$. Construct a program to count the number of integers k and l satisfying

$$0 \leq k \leq N \ \wedge\ 0 \leq l \leq M \ \wedge\ M \times k = N \times l \ .$$

The running time of your program should be linear in $M+N$. (Hint: the function mapping k and l to $M \times k - N \times l$ is strictly increasing in k and strictly decreasing in l . Develop a suitably modified implementation of saddleback search.)

State precisely the postcondition of your program as well as the invariant property and bound function on which any loop in your program is based. Justify clearly every test and assignment in your program. (15)

(b) Modify the program so that all multiplications (eg. when evaluating $M \times k$) are replaced by additions. Make clear how any assertions in your program are modified as well. (10)

4 The simple graph-searching algorithm below was developed in the lectures. N is the set of nodes and $directlyreachable$ maps a node u into the set of nodes reachable by a single edge from u . The algorithm computes $reachable.\{s\}$ where $reachable.M$ is the set of nodes reachable by a path in the graph (of edge-count 0 or more) that starts in a node in the set M .

```
{  $s \in N$  }
$black , grey , white \ := \ \emptyset , \{s\} , N - \{s\}$ ;
{  **Invariant:** $reachable.\{s\} = black \cup reachable.grey \ \wedge\ \{black, grey, white\}$ partitions $N$
   **Bound function:** $|black|$  }
while $grey \neq \emptyset$ do
   begin
      choose $u \in grey$; add $u$ to $black$ and remove it from $grey$;
      for $v \in directlyreachable.u$
      do if $v \in white$
           then add $v$ to $grey$ and remove it from $white$
   end
{  $reachable.\{s\} = black$  }
```

a) Explain in words how the invariant makes precise the function of the three sets $black$, $grey$ and $white$. (5)

b) Discuss data structures that might be used to implement the sets $black$, $grey$ and $white$. Make clear what the basis is for your choice of data structures. (10)

c) Consider a modification of the algorithm whereby a finish node f is specified. The task is to find a path of smallest edge count from s to f . (You may assume that such a path exists.) Make the appropriate changes to the algorithm, making sure you specify clearly the invariant property maintained by your algorithm. Discuss the effect this has on the choice of data structures. (10)

5 Given is a one-dimensional array A of (real) numbers of length N (a natural number at least 0). The array is indexed by numbers i where $0 \leq i < N$. Using dummy S to range over sets of indices, consider the problem of calculating

$$\langle \Uparrow S : S \text{ nc } N : \text{sum}.S \rangle$$

where

$$[\ S \text{ nc } N \ \equiv\ S \subseteq \{i \mid 0 \leq i < N\} \wedge \langle \forall i :: i \notin S \vee i{+}1 \notin S \rangle\ ]$$

and

$$[\ \text{sum}.S\ =\ \langle \Sigma i : i \in S : A[i] \rangle\ ]\quad .$$

A first step in the solution of such a problem is to express the constraint nc inductively. As discussed in the lectures, its inductive definition is given by:

(5.1) $[\ S \text{ nc } (-1)\ \equiv\ S = \emptyset\ ]$

(5.2) $[\ S \text{ nc } 0\ \equiv\ S = \emptyset\ ]$

and

(5.3) $[\ S \text{ nc } k{+}1\ \equiv\ S \text{ nc } k\ \vee\ \langle \exists T : S = T \cup \{k\} : T \text{ nc } k{-}1 \rangle\ ]\quad .$

Explain how to construct a graph that represents all sets S such that S nc k (for a given k) as a set of paths. Show your construction for the case that k is 5. Explain how the edge labels in the graph correspond to the terms in (5.3). (10)

Suppose the function opt is defined by

$$[\ \text{opt}.k\ =\ \langle \Uparrow S : S \text{ nc } k : \text{sum}.S \rangle\ ]\quad .$$

Using (5.3), prove that

(5.4) $[\ \text{opt}.(k{+}1)\ =\ \text{opt}.k \uparrow (\text{opt}.(k{-}1) + A[k])\ ]\quad .$

Make clear all steps in your calculation by providing suitable hints. (10)

Derive formulae for opt.0 and opt.(-1). Explain how the solution of the equations you have derived together with (5.4) is represented as finding a longest path in a graph. (5)