

**Due:** By 6pm on Thursday 22 November.

**Worth:** 15%

This assignment is to be completed in groups of no more than three students. Hand in a single paper for your group, with the information about each student filled in on the cover page.

Keep in mind that your proofs will be graded on their structure at least as much as on their content. In other words, when we ask you to prove a statement, it is to find out how well you can write proofs, not because we are interested in knowing whether or not that statement is true. So pay particular attention to writing your proofs properly.

**GENERAL ADVICE:** I know many people split up the work on group assignments so that each person is “responsible” for one or two questions only. But this assignment is complicated enough that I believe you would greatly benefit from working on it together with your partner(s) rather than trying to solve it individually. Remember that this is supposed to be the point of these group assignments: to give you a chance to work on problems together, so that you each learn more than by doing it yourself.

Also, splitting up the work may save time in the short term, but not in the long term: since everyone is expected to understand how to solve each question, you will each have to go back and review each solution anyway. More importantly, as you well know, there is a big difference between reading someone else’s solution, and working out a solution by yourself: you learn much more by “solving” than by “reading”.

Just keep it in mind. . .

1. [5 marks]

Prove that the following algorithm terminates.

```
# Pre:  $x \in \mathbb{N}$ 
 $y = x * x$ 
while  $y \neq 0$ :
     $x = x - 1$ 
     $y = y - 2 * x - 1$ 
```

**HINT:** Derive and prove a loop invariant whose purpose is to help prove termination.

2. [10 marks]

Consider the following variation on Binary Search, where  $n = \text{len}(A)$ .

```
# Pre:  $A[0 \dots n - 1]$  is sorted and  $x$  is comparable with elements of  $A[0 \dots n - 1]$ 
 $b = 0$ 
 $e = n$ 
# LI:  $A[0 \dots b - 1] \leq x < A[e \dots n - 1]$  and _____
while _____:
     $m = (b + e) / 2$  # integer division
    ...
# Post:  $0 \leq p \leq n$  and  $A[0 \dots p - 1] \leq x < A[p \dots n - 1]$ 
```

Complete the loop invariant with suitable conditions on the values of  $b$  and  $e$ , then complete the code to satisfy the postcondition and loop invariant. Explain your reasoning, *i.e.*, explain how you are using the LI to help you write the code, and how the code is dictating parts of the LI (using pictures is fine). Then, justify that your algorithm is correct (including termination)—this means you don’t need to write a formal proof, but you should provide the main arguments that would go in such a proof.

3. [15 marks]

Prove that the algorithm on the next page is correct. This is a complex algorithm, so take it step-by-step! In particular, draw some pictures to help you figure out what’s going on and to keep track of how values change during execution.

```

# Pre:  $0 \leq b \leq e < \text{len}(A)$ ,  $A[b \dots e]$  is a list of comparable elements,
#       and  $k$  is an integer such that  $1 \leq k \leq e - b + 1$ 
# Post:  $\text{EXTRACT}(k, A, b, e)$  returns the value of the  $k$ -th smallest element in  $A[b \dots e]$ 
EXTRACT( $k, A, b, e$ ):
    if  $e == b$ : return  $A[b]$ 
     $c = b + 1$ 
    while  $c \leq e$  and  $A[c] \leq A[b]$ :  $c = c + 1$ 
     $d = e$ 
    while  $A[d] > A[b]$ :  $d = d - 1$ 
    while  $c \leq d$ :
         $A[d], A[c] = A[c], A[d]$  # swap
        while  $A[c] \leq A[b]$ :  $c = c + 1$ 
        while  $A[d] > A[b]$ :  $d = d - 1$ 
     $A[d], A[b] = A[b], A[d]$  # swap
    if  $k < d - b + 1$ : return  $\text{EXTRACT}(k, A, b, d - 1)$ 
    if  $k > d - b + 1$ : return  $\text{EXTRACT}(k - d + b - 1, A, d + 1, e)$ 
    return  $A[d]$ 

```

4. [12 marks]

For each language  $L_i$  below, give a regular expression  $R_i$  such that  $L(R_i) = L_i$  and a finite state automaton  $A_i$  such that  $L(A_i) = L_i$ . Justify briefly that your R.E. and your F.S.A. are correct (*i.e.*, that they correspond exactly to the language  $L_i$ ).

(a) [4 marks]

$L_1 = \{s \in \{a, b, d\}^* : s \text{ contains the substring } dab \}$

For example,  $bdb \notin L_1$ ,  $bdab \in L_1$ ,  $badb \notin L_1$ ,  $badab \in L_1$ ,  $badaab \notin L_1$ .  $baadab \in L_1$ .  $baadaab \notin L_1$ .

(b) [4 marks]

$L_2 = \{s \in \{a, b, d\}^* : s \text{ does not contain the substring } bad \}$

For example,  $bdb \in L_2$ ,  $bdab \in L_2$ ,  $badb \notin L_2$ ,  $badab \notin L_2$ ,  $badaab \notin L_2$ .  $baadab \in L_2$ .  $baadaab \in L_2$ .

(c) [4 marks]

$L_3 = \{s \in \{a, b, d\}^* : s \text{ contains the substring } dab \text{ and } s \text{ does not contain the substring } bad \}$

For example,  $bdb \notin L_3$ ,  $bdab \in L_3$ ,  $badb \notin L_3$ ,  $badab \notin L_3$ ,  $badaab \notin L_3$ .  $baadab \in L_3$ .  $baadaab \notin L_3$ .

5. [3 marks]

Prove that all finite languages can be described by a regular expression, *i.e.*, for any finite set of strings  $L$ , there exists a regular expression  $R$  such that  $L(R) = L$ .