IMPERIAL COLLEGE OF SCIENCE, TECHNOLOGY AND MEDICINE

EXAMINATIONS 2023

BEng Honours Degree in Computing Part II
MEng Honours Degrees in Computing Part II
BEng Honours Degree in Mathematics and Computer Science Part II
MEng Honours Degree in Mathematics and Computer Science Part II
BEng Honours Degree in Mathematics and Computer Science Part III
MEng Honours Degree in Mathematics and Computer Science Part III
for Internal Students of the Imperial College of Science, Technology and Medicine

This paper is also taken for the relevant examinations for the Associateship of the City and Guilds of London Institute

PAPER COMP50006

COMPILERS

Friday 5th May 2023, 10:00
Duration: 90 minutes

Answer ALL TWO questions

Paper contains 2 questions
Calculators not required

1a Consider the following grammar

```
E → C C
C → a C | b
```

For this grammar draw the DFA of LR(1) items.

b For the grammar in part a

i) Construct the parse table for the DFA of LR(1) items.

ii) What states must be merged to derive the DFA of LALR(1) items? You do not need to draw the DFA of LALR(1) items.

c For the remaining parts of this question consider a C-like language that supports two-dimensional arrays (matrices).

Give 4 checks that the semantic analyser should do for matrix declarations like in the following:

```
int m1[5,8];
int m2[8,5];
```

Note: array sizes are integer values, not integer expressions.

d Give 4 checks that the semantic analyser should do for a sub-matrix assignment like in the following:

```
int a, b, c, d, e, f, g, h;
...
m1[a..b, c..d] = m2[e..f, g..h];
```

where the range `expr1..expr2` selects rows `expr1` to `expr2` when used for the 1st dimension of the matrix and columns `expr1` to `expr2` when used for the 2nd dimension of the matrix.

e Map the sub-matrix assignment in part d to equivalent C-like code without ranges. Your code must perform array bounds checking.

The five parts carry, respectively, 30%, 20%, 10%, 10%, 30% of the marks.

2 a Consider code generation for a processor whose register usage convention specifies that some registers are *callee saves* while other registers are *caller-saves*. Under what circumstances should the compiler choose to allocate a (register-allocatable) variable to a register in the *caller-saves* set?

For the remainder of this question, consider the following program fragment:

```
S1:     int A[1000];
S2:     int f(int n) {
S3:       int s = 0;
S4:       int r = 0;
S5:       int x;
S6:       int i = 0;
S7:       while (true) {
S8:         if (i>n) break;
S9:         r = r+s;
S10:        x = A[i];
S11:        if (x==0) {
S12:          s = n/3;
            } else {
S13:          s = n/5;
            }
S14:        i = i+1;
          }
S15:      return r;
        }
```

b Which of the variables `n`, `s`, `r`, `x` and `i` are *induction* variables? Explain your answer briefly.

c How many *natural loops* are there in this program fragment's control flow graph? Explain your answer briefly.

d Which of the definitions (S2, S3, S4, S5, S6, S9, S10, S12, S13, S14) are the *relevant* reaching definitions for `S9`? Explain your answer briefly.

e Why are definitions S12 and S13 *candidates* for loop-invariant code motion? Explain your answer briefly.

f It is not safe simply to move S12 and S13 to the loop's pre-header. Give at least *two* reasons why; explain your answer carefully.

g If we transform this code fragment (S3-S15) into SSA (Static Single Assignment) form, we need to introduce some "ϕ" ("phi") nodes. What phi nodes are needed for variable `s`, and before which lines (S3-S15) should they be placed?

The seven parts carry, respectively, 15%, 10%, 10%, 10%, 15%, 20%, and 20% of the marks.