Spring, 2023 Name: _____

(Please, do *not* write in your id number!)

COP 3402 — Systems Software

Midterm Exam

Directions for this Test

This test has 8 questions and pages numbered 1 through 6.

This test will be for the entire period in class and is be closed book.

However, you may use one (1) page of notes on one (1) side of a standard 8.5 by 11 inch sheet of paper. These notes can either be hand-written or printed, but if printed, then the font must be a 9-point or larger font. These notes must be turned in with the exam.

If you need more space, use the back of a page. Note when you do that on the front.

Before you begin, please take a moment to look over the entire test so that you can budget your time.

For Grading

Question:	1	2	3	4	5	6	7	8	Total
Points:	15	10	10	10	10	10	15	20	100
Score:									

All questions on this exam are related to the course outcome [Concepts].

1. (15 points) Which of the following are "systems software"? (Circle **each** answer letter that is correct.)

   A. A networked music player (like Spotify).

   B. A movie player (like VLC).

   C. A compiler (like gcc).

   D. A photo editor (like Photoshop).

   E. A computer-aided design program (like AutoCAD)

   F. A linker or loader (like ld).

   G. A spreadsheet application (like Excel).

   H. An assembler (like gas).

   I. A social network (like Facebook).

   J. An operating system (like Linux).

2. (10 points) What happens to a VM's PC register if no halt (HLT) instruction is executed? (Circle the **one** correct answer's letter.)

   A. The VM stops when the PC reaches program's last instruction.

   B. The VM notices that the program is done and halts.

   C. The VM interprets each address as an instruction, and keeps adding 1 to the PC.

   D. The VM is not supposed to stop, so it issues an error message when the PC reaches the program's last instruction.

   E. The VM takes a holiday and does nothing.

3. (10 points) Fill in the blanks in the following.

   The two components of a lexical address are:

   1. the _____, and
   2. the _____ .

4. (10 points) Consider the following PL/0 program.

```
const b = 32, n = 10, r = 13;
var i, k;
procedure F;
  var k, l, m;
  begin
    read k;    # asking about k here
    q := k
  end;
procedure G;
 write n;
begin
  read i;
  write i
end.
```

   Assuming that the VM is word-addressed and that constants and variables are allocated in the order in which they are declared, what is the lexical address of the variable `k` that is used on the line indicated? (Circle the **one** correct answer's letter.)

   A. $(3, 2)$
   B. $(3, 1)$
   C. $(3, 0)$
   D. $(1, 3)$
   E. $(1, 2)$
   F. $(1, 1)$
   G. $(1, 0)$
   H. $(0, 4)$
   I. $(0, 3)$
   J. $(0, 2)$
   K. $(0, 1)$
   L. $(0, 0)$

5. (10 points) Consider again the following PL/0 program.

```
const b = 32, n = 10, r = 13;
var i, k;
procedure F;
  var k, l, m;
  begin
    read k;
    q := k
  end;
procedure G;
  write n;
begin
  read i; # asking about i here
  write i
end.
```

Assuming that the VM is word-addressed and that constants and variables are allocated in the order in which they are declared, what is the lexical address of the variable `i` that is used on the line indicated? (Circle the **one** correct answer's letter.)

A. $(3, 2)$

B. $(3, 1)$

C. $(3, 0)$

D. $(1, 3)$

E. $(1, 2)$

F. $(1, 1)$

G. $(1, 0)$

H. $(0, 4)$

I. $(0, 3)$

J. $(0, 2)$

K. $(0, 1)$

L. $(0, 0)$

6. (10 points) Consider a programming language with reserved words: choose the option below that describes the correct way to recognize reserved words as tokens in a lexical analyzer with ASCII character input. (Circle the **one** correct answer's letter.)

   A. When an implementation finds an input text that is more shy and reserved than other texts, it returns that as a reserved word.

   B. When the lexical analyzer finds an input text that is a short, lower case English word, then it returns that as a reserved word.

   C. When an implementation finds an input text that matches the grammar for an identifier, it compares that text against the text of the reserved words, and returns a token for the corresponding reserved word whenever it finds a match.

   D. When an implementation finds an input text that is fundamental to the program, then it returns the corresponding reserved word.

   E. When the lexical analyzer finds an input text that is in bold font, then it returns that as a reserved word.

7. (15 points) Consider the following grammar in BNF notation:

   ⟨term⟩ `::=` ⟨ident⟩ | `(` **`fun`** ⟨ident⟩ `.` ⟨term⟩ `)` | `(` ⟨term⟩ ⟨term⟩ `)`

   where `f`, `x`, `y`, and `z` are ⟨ident⟩s. Which of the following are legal ⟨term⟩s according to this grammar? (Circle **each** answer letter that is correct.)

   A. `((`**`fun`** `z.(z z)) (`**`fun`** `z.(z z)))`

   B. `(`**`fun`** `y.y)`

   C. `x y x`

   D. `(`**`fun`** `(y) (f y 2))`

   E. `x`

   F. `3402`

8. (20 points) This problem is about writing a recognizer in C for the following production:

⟨def⟩ ::= **define** ⟨ident⟩ ⟨expr⟩

Assume there is an included header file `token.h` that defines an **enum**, `token_type`, which includes `definesym` for the reserved word **define** and `identsym` as for an ⟨ident⟩. Assume that your group has already written a function to recognize an ⟨expr⟩:

```
extern void parseExpr();
```

Assume also that they have defined a variable `tok` and a function `eat`, as we have done in class (where `eat(tt)` checks that `tok.typ` is `tt` and if so, then it advances the lexer and assigns the new token to `tok`, and otherwise issues an error message and exits the program):

```
static token tok;
static void eat(token_type tt);
```

Which of the following is the correct code for a C function `parseDef` that parses the above production in a recursive descent parser? (Circle the **one** correct answer's letter.)

A.
```
void parseDef() {
  switch (tok.typ) {
  case definesym:
    switch (tok.typ) {
    case identsym:
      parseExpr(); break;
    default: break;
    }
  default: break;
  }
}
```

B.
```
void parseDef() {
  if (strcmp(tok, "define") == 0) {
    if (strcmp(tok, "ident") == 0) {
      parseExpr();
    } else { /* report error ... */
  } else { /* report error ... */ }
}
```

C.
```
void parseDef() {
  eat(definesym);
  eat(identsym);
  parseExpr();
}
```

D.
```
void parseDef() {
  strcmp(tok, "define");
  eat(identsym);
  parseExpr();
}
```