CS3432 Computer Organization
Spring 2025. Midterm I
03/06/2025 – 4:30PM to 5:50PM MST / to 03/07/2025 1:59AM MST

All documents allowed - Internet access NOT allowed for the theoretical part

You need to turn in the theoretical part of the exam (Sections 1 and 2) by 03/06/2025 5:50PM MST on paper, directly to your instructor. You need to turn in the practical part of the exam (Section 3) by 03/07/2025 1:59AM MST by email to `utep-spring-2025-arc-midtermI@christoph-lauter.org`.

1 Reading and Testing Code

Imagine you work in a company that reuses C code that has been written a while ago. When you first open the code, you find a function `strsprt` defined by 27 lines of code which, unfortunately, are slightly cryptic[1]:

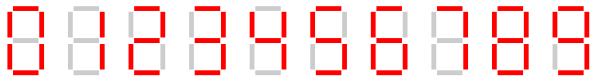
```
size_t strsprt(char **tks, char *s) {
  size_t n;
  char *c;
  char *tk;

  n = (size_t) 0;
  c = s;
  for (tk=c; *tk=='␣'; tk++);
  c=tk;
  while (*c!='\0') {
    if (*c=='␣') {
      tks[n] = tk;
      n++;
      *c='\0';
      c++;
      for (tk=c; *tk=='␣'; tk++);
      c=tk;
    } else {
      c++;
    }
  }
  if (*tk!='\0') {
    tks[n] = tk;
    n++;
  }
  return n;
}
```

[1] In the listing, '␣' stands for the space character. In C, **char** `**a` declares a variable `a` that is a pointer to a pointer to a **char** variable.

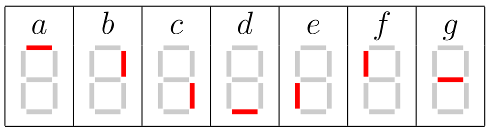
- Explain what that function `strsprt` does. Give example input arguments and make a drawing with arrows indicating the different pointers. Use two colors or make two drawings to show the state of the computer's memory, i.e. the pointers, before and after the function `strsprt` runs on your example inputs.
- **Do not paraphrase the code**, i.e. do not say things like ... *and then, the* `c` *variable is incremented by 1* ... but rather say things like ... *the function takes* ... *in argument and converts that argument into several*....
- Now that you have explained the code in function `strsprt`, give example calls to the function, **together with example argument values**, such that each line of the code is executed at least once and its purpose is made clear. State what each example call does and why this is the case.

2 Seven Segments

Small little, cheap computer-controlled devices often display numbers on so-called seven segment LED displays. These displays contain seven LEDs[2], often of red or green color, arranged in such a way that by lightning them up in certain patterns, the decimal digits 0 through 9 can be displayed:



These seven segments are commonly labeled a, b, c, d, e, f, g and they connected to the seven LEDs as shown here:



In order to translate from a binary number n in the range 0 to 9, given with its four bits n_3, n_2, n_1, n_0, to the right segments, so-called *seven segment decoders* are sold as Integrated Circuits. These seven segment decoders take the four bits n_3, n_2, n_1, n_0 in input and produce seven outputs a, b, c, d, e, f, g. Each output is set to 1 if the corresponding segment needs to be lit to show the digit corresponding to the number fed into the circuit on inputs n_3 through n_0. For example, for input $n = 5$, i.e. $n_3 = 0, n_2 = 1, n_1 = 0, n_0 = 1$, the g output is $g = 1$ but the b output is $b = 0$, as the digit 5 requires the g segment to be lit but the b segment to be switched off.

- Give a truth table with four inputs n_3, n_2, n_1 and n_0 and seven outputs a, b, c, d, e, f and g specifying what a seven segment decoder circuit needs to implement.
- Now pick **two** output columns of your choice, **indicate which two columns you picked**, and give two circuits implementing these two columns, based on NAND gates, which have two inputs and one output. You will end up with a circuit for example for the a column and a circuit for example for the g column.

  In the case your circuits need to use AND gates with k inputs and one output or OR gates with m inputs and one output, you first need to give implementations of these circuits with NAND gates, which have two inputs and one output. You can then instantiate any number of these AND or OR gates that you need.

[2]small lamps

3 Dictionary (Practical Part)

For this Section, you need to complete the missing parts in the boilerplate code `dictionary.c` that is given on the course website. Test your program with at least the example runs given in comments.

Here are example runs of the completed dictionary program:

```
$ ./dictionary
Please enter a word in Spanish (lower case, no accents, no tildas): madre
The spanish word "madre" you entered means "mother" in English.
$ ./dictionary
Please enter a word in Spanish (lower case, no accents, no tildas): padre
The spanish word "padre" you entered means "parent" in English.
$ ./dictionary
Please enter a word in Spanish (lower case, no accents, no tildas): abajo
The spanish word "abajo" you entered means "below" in English.
$ ./dictionary
Please enter a word in Spanish (lower case, no accents, no tildas): zona
The spanish word "zona" you entered means "zone" in English.
$ ./dictionary
Please enter a word in Spanish (lower case, no accents, no tildas): wurst
The spanish word "wurst" you entered could not be found in the dictionary.
```

Please be aware that submissions of code that does not compile will receive very little credit. Please also take into account that submissions of code where the `lookup` function, which you need to implement, has linear complexity instead of the required logarithmic complexity in $\mathcal{O}(\log n)$ will receive very little credit for that function.