CISS245: Advanced Programming
Quiz q2701

Name: YOUR EMAIL Score: □

Q1. In the answer box below, allocate memory for `p` and `q` to point to:
(a) write one statement so that `p` points to an array of size 10 in the heap and
(b) write one statement so that `q` points to the third value (i.e., index 2 value) of the array that `p` points to.
Answer:

```
double *p, *q;

```

Q2. The function below has memory issues. Fix the problem.
Answer:

```
void f()
{
    double * p = new double;
    char * q = new char[1024];
    // Write two statements below to fix the memory issues

    return;
}
```

Q3. There is an array with values 2, 3, 5, 7, 11 in the heap. A pointer `p` is pointing to the value 7 of this array. Complete the statement below so that `q` points to the value 3 in the above array.
Answer:

```
int * q =          ;
```

Q4. Complete the following linear search function so that if `p` is returned, then `p` points to the first time `target` appears in the array with beginning address `start` and ending address `end - 1`. If `target` is not found, then `NULL` is returned.
Answer:

```
int * linearsearch(int * start, int * end, int target)
{
    for (int * p =          ; p <          ; ++p)
    {
        if (              )
        {
            return            ;
        }
    }
    return              ;
}
```

(This should be a straightforward translation of the linear search algorithm from CISS240. Do not change the algorithm other than translating it from array indexing to pointer scanning.)

Instructions

In the file `thispreamble.tex` look for

```
\renewcommand\AUTHOR{}
```

and enter your email address:

```
\renewcommand\AUTHOR{jdoe5@cougars.ccis.edu}
```

(This is not really necessary since alex will change that for you when you execute `make`.) In your bash shell, execute “`make`” to recompile `main.pdf`. Execute “`make v`” to view `main.pdf`.

Enter your answers in `main.tex`. In the bash shell, execute “`make`” to recompile `main.pdf`. Execute “`make v`” to view `main.pdf`.

For each question, you'll see boxes for you to fill. For small boxes, if you see

```
1 + 1 = \answerbox{}.
```

you do this:

```
1 + 1 = \answerbox{2}.
```

`answerbox` will also appear in “true/false” and “multiple-choice” questions.

For longer answers that need typewriter font, if you see

```
Write a C++ statement that declares an integer variable name x.
\begin{answercode}
\end{answercode}
```

you do this:

```
Write a C++ statement that declares an integer variable name x.
\begin{answercode}
int x;
\end{answercode}
```

`answercode` will appear in questions asking for code, algorithm, and program output. In this case, indentation and spacing is significant. For program output, I do look at spaces and newlines.

For long answers (not in typewriter font) if you see

```
What is the color of the sky?
\begin{answerlong}
\end{answerlong}
```

you can write

```
What is the color of the sky?
\begin{answerlong}
The color of the sky is blue.
\end{answerlong}
```

A question that begins with “T or F or M” requires you to identify whether it is true or false, or meaningless. “Meaningless” means something’s wrong with the question and it is not well-defined. Something like “$1 + 2 = 4$” is either true or false (of course it’s false). Something like “$1+_2 = 4$?” does not make sense.

When writing results of computations, make sure it’s simplified. For instance write 2 instead of $1 + 1$.

Higher level classes.

For students beyond 245: You can put LaTeX commands in `answerlong`.

More examples of meaningless statements: Questions such as “Is $42 = 1+_2$ true or false?” or “Is $42 = \{2\}^{\{3\}}$ true or false?” does not make sense. “Is $P(42) = \{42\}$ true or false?” is meaningless because $P(X)$ is only defined if X is a set. For “Is $1 + 2 + 3$ true or false?”, “$1 + 2 + 3$” is well–defined but as a “numerical expression”, not as a “proposition”, i.e., it cannot be true or false. Therefore “Is $1 + 2 + 3$ true or false?” is also not a well-defined question.

More examples of simplification: When you write down sets, if the answer is $\{1\}$, do not write $\{1, 1\}$. And when the values can be ordered, write the elements of the set in ascending order. When writing polynomials, begin with the highest degree term.

When writing a counterexample, always write the simplest.