CISS350: Data Structures and Advanced Algorithms
Quiz q05

Name: YOUR EMAIL

Score:

Q1. Complete the following recursive function to compute GCD. (Yes I know this is straight from my notes. Do it anyway.)

Answer:

```
// Do NOT use for-loops or while-loops or do-while-loops
int gcd(int m, int n)
{
}
```

TURN PAGE

Q2. Complete the following recursive function `print(m, n)` that prints a sequence of consecutive integers starting with `m` and ending with `n - 1`. For instance the output of

```
print(2, 6);   // prints 2 3 4 5 and newline
print(10, 18); // prints 10 11 12 13 14 15 16 17 and newline
```

is

```
2 3 4 5
10 11 12 13 14 15 16 17
```

Note that the function must be recursive.

Answer:

```
// Do NOT use for-loops or while-loops or do-while-loops
void print(int m, int n)
{
    if (m >= n)
    {
                                 // <-- Write ONE statement here
    }
    else
    {
                                 // <-- Write ONE statement here
        print(m + 1, n);         // <-- Do not change this statement
    }
}
```

TURN PAGE

Q3. Complete the following recursive function `int count(int *start, int *end, int target)` that counts the number of times `target` appears in `*start`, `*(start + 1)`, `*(start + 2)`, ..., `*(end - 1)`. For instance the output of

```
int x[] = {1, 2, 3, 4, 3, 2, 1, 2, 3, 4};
std::cout << count(x + 2, x + 8, 3) << '\n';
```

is 2. Note that the function must be recursive.
Answer:

```
// Do NOT use for-loops or while-loops or do-while-loops
int count(int * start, int * end, int target)
{
    if (start >= end)
    {
        return 0;
    }
    else
    {

    }
}
```

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.