CISS240: Introduction to Programming
Quiz q0602

Name: bglandis1@cougars.ccis.edu Score:

This is a closed-book, no compiler, 5 minute quiz.

Q1. Write down the output of the following code fragment:

```
int x = 0, y = 1, z = 2;
std::cout << (x < y && x + y > z) << std::endl;
```

Answer:

```
0
```

Q2. If `x`, `y`, `z` are integer variables with values 0, 1, 2 (respecively), what is the boolean value of the follow expression:

```
x > y + z || x + y < 3 * z && x > y
```

Answer:

```
false
```

Q3. Simplify the following boolean expression (do not evaluate - because you can't):

```
!(x < y) && (x >= w || w <= x) || ((x < 2) == true)
```

Answer:

```
true
```

Q4. Complete the following code fragment. It prompts the user for the following values:

- amount in the user's savings account (stored in variable `savings`)
- monthly rent (stored in variable `monthly_rent`)
- daily expenses (stored in variable `daily_expenses`)
- number of days in the current month (stored in variable `days_in_month`)

You must initialize the boolean variable `call_home_for_cash` so that it is set to true exactly when the amount in the user's savings account is less than the total expenses for the current month. The total expenses is (obviously) the sum of the following

- monthly rent and product of daily expenses and number of days in the current month

Answer:

```
int savings, monthly_rent, daily_expenses, days_in_month;
std::cin >> savings >> monthly_rent >> daily_expenses >> days_in_month;
bool call_home_for_cash = savings < (monthly_rent + daily_expenses * days_in_month);
std::cout << call_home_for_cash << std::endl;
```

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.