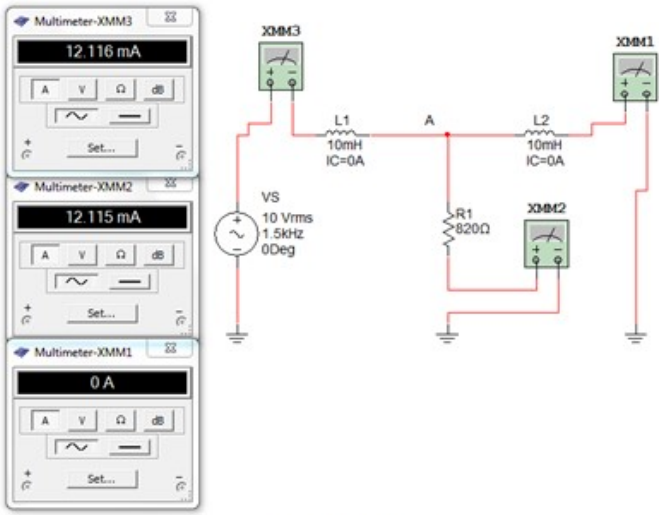


Figure 2: Simulated circuit

Back

Computer Science

List of changes made over Version 9.1

- Modified Subject specific guidelines of CSE……Page no. 84

List of changes made over Version 9.2

- Removed points (e), (f) and (g) from CS1.Content second point of Subject specific guidelines of CSE……Page no. 85

List of changes made over Version 9.3

- 4th bullet point of Fourth part of 2.a is modified……Page no.87
- 5th bullet point of fourth part of 2.a is completely changed……..Page no.88

Table of Contents

CS1. Content

1. Points to be note while working on Authoring/QA:

a. Read and understand the question carefully. Identify all the requirements in the question/problem.

b. Decide the divisions in the solutions such that each division is a meaningful and complete step.

c. Author the solution as per the divisions identified.

d. Give proper conclusion at the end of the solution.

e. Solution should be conceptually 100% accurate.

f. Solution should be complete in all aspects of the question.

g. Solution should be within the context of the question.

h. Use only the topics so far covered in the textbook to author the solution. Should not use the topics discussed later in the textbook.

i. Presentation should be user friendly and easy to understand.

j. Use Serif Draw plus (or any other equivalent software) to draw flowcharts, UML diagrams, diagrams, edit screen shots etc.

- The images should be clearly visible and readable including subparts.

k. Use short, simple and grammatically correct sentences in your answers.

l. Give explanation point-by-point using bullet points. Avoid writing paragraphs.

m. Use simple sentences for explanation. Avoid complex and ambiguous sentences. Avoid unnecessary explanation.

n. While authoring programming solutions use the naming conventions as per the textbook in the program. If the textbook naming conventions are not available, then use the following naming conventions:

1. *Class or Interface Names*: Use a noun or combination of nouns which reflects the behaviour of the class. Use uppercase character as starting character for each noun (if name contains multiple nouns). For example “FahrenheitPanel” can be a valid interface name to represent a Panel of a Fahrenheit. Don’t separate words using space or hyphen.
2. *Method, Attribute and Local Variable names*: Use a verb, noun or combination both which reflects what exactly the method does. Use lowercase character as

starting character for first word and uppercase character for subsequent words. For example “arcAngle” can be a valid variable name to represent “angle of an arc”.

3. *Constant names:* Use all uppercase characters for constants. For example “MAXVALUE”.

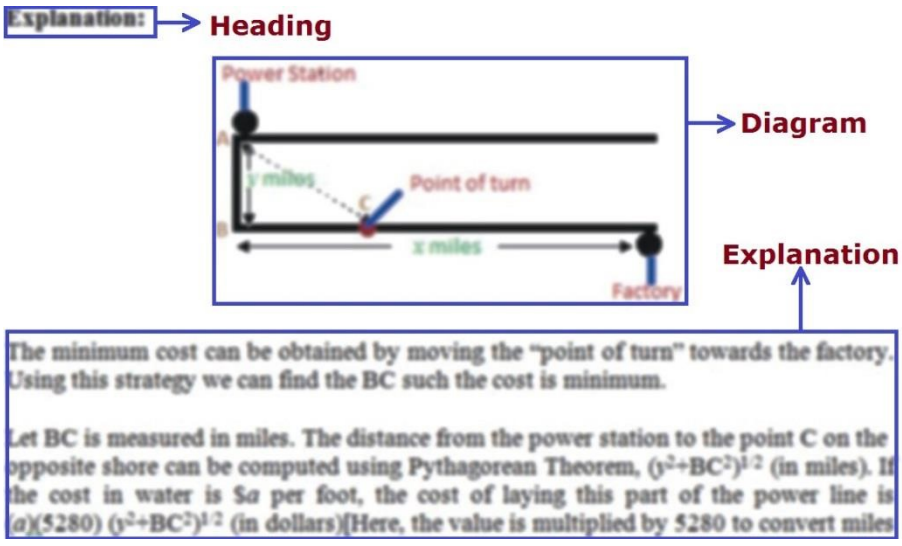
o. Code should be presented in default colours of the specified IDE.

2. Guidelines to author solutions for different types of questions:

a. Programming type questions:

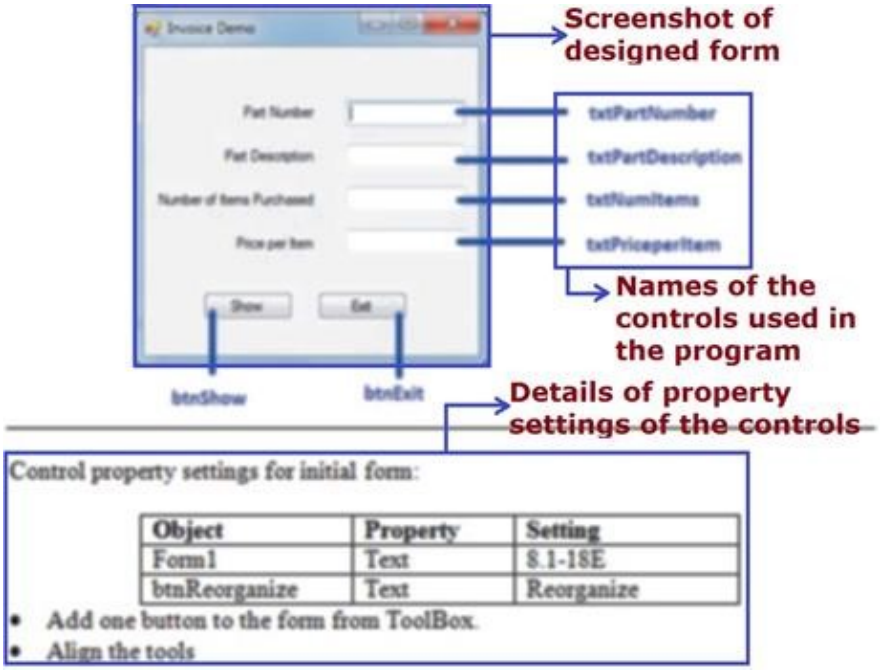
While authoring solutions related to programming, there are multiple logical parts, with each part having one or more Chegg steps.

- *First part (needed only in special cases):* If the situation given is complex or involve mathematics or topics from other subjects; then the analysis of the situation/problem should be explained in this part. Use diagram(s) if needed. Divide into one or more Chegg steps as per the requirement for better understanding.

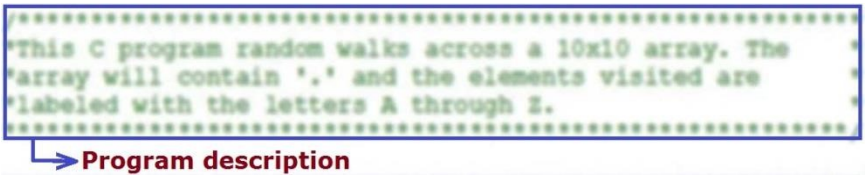


- *Second part:* Give explanation briefly, how the program will be implemented and how it works under the heading “**Program Plan**:”

- *Third part (needed only in special cases):* When designing forms in “.net” environment (or any equivalent environment), the steps involved in placing the controls, naming etc. should be explained.



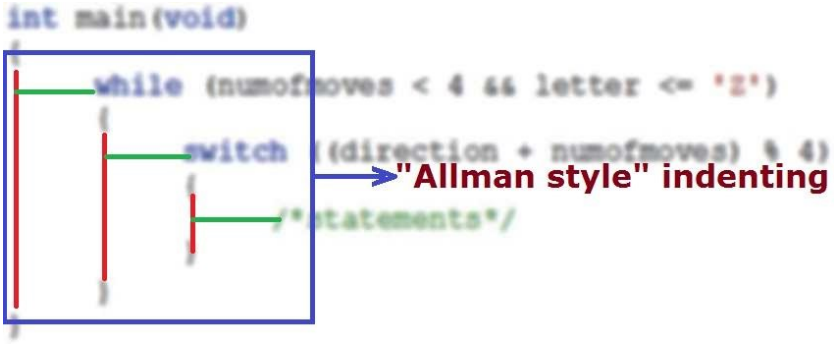
- *Fourth part:* Write the program code in specified programming language under the heading "**Program**:". The code should be formatted as follows:
  - In the beginning of the program use comment block. In that block, write description (in one or two lines) what the program does.



  - If any specific header files or import packages are used (other than usual), then write details about the functions or methods used from those header files or packages.
  - Before function/method heading, explain their functionality in comments.
  - Write necessary comments to understand the statements and the logical flow of the program
  - Write comments for every statement in the function, method, and definitions.

- Use a delimiter to separate functions (C Language) and different classes (Object Oriented Languages). Give explanation in comments about the function/class in one or two lines. Don't break the function or class using a delimiter in the middle.
- Write the program using the topics/concepts covered so far in the textbook. Should not use the topics/concepts that are covered in the later chapters.

- The program should be indented properly. The indenting should follow "Allman Style".



- *Fifth Part (needed only in special cases):* Write this part in the following situations with appropriate heading.
  - If the program requires any special execution method (example: command line arguments in C or C++), the procedure of execution should be explained.
  - If the program needs to work with database, give step-by-step procedure to run the program.

- *Sixth part:* The output is given under the heading "**Sample Output**:". Present the output as follows:
  - Show all the important outputs of the program.
  - If the program is console-based, then the output should be in text form directly copied from the console (using Clipboard pasting).
  - If the program is GUI based, then the output should contain screenshots.

- *Seventh part (needed only in special cases):* Include this part if the output needs to be explained.

Note: Use specified Integrated Development Environments (IDEs) for compiling, debugging, and running the programs. Present the code in colors as per the default colors of the IDE.

b. Write function/method:

- Write the function/method in the user specified language, within the scope of the concepts, and functionality specifications.
- Write the method with the parameters and return values as specified.
- Use adequate comments to understand the statements, logic, and functionality.
- Try to demonstrate the functionality of the function/method using a simple driver program (If the function/method is independent and can be called directly).

c. Write specific parts of the program/Modifying the program:

- Identify the statement or method need to be added.
- Identify the location in the program where the statement or method needs to be added.
- Don't write complete program given in the textbook. Instead give reference to the location in the textbook where the code is available. Use Figure No., Table No., Listing No., as references. Avoid using page numbers as references.
- Highlight the code that is added/modified using "Text Highlighter" in grey colour. It helps the user to identify the additions/modifications easily.
- Give the code (skeleton) with the required additions.
- Present the code with proper indentation along with output.

d. Rectify the errors in the code/statements:

- Read and understand the question carefully and identify the statements with errors.
- Give explanation, why the particular statement/code is an error.
- Fix all the errors in the program/code/statement and make the code executable if possible.
- If program is given to rectify errors, then highlight the code that is rectified using "Text Highlighter" in grey colour. It helps the user to identify the rectifications easily and give the code (skeleton).

  (*Refer guideline 2(c) for more details on presentation.*)

e. Differences between two concepts:

- Use a table to give the differences.
- Give as many (at least 5) differences as possible.
- Distinguish with the help of detailed explanation.

- If the question asked to compare and contrast, give the comparisons first and differences next (don't mix).

f. Matching questions:

- Identify the correct matching terms and give the appropriate matching using a table.
- Give simple explanation stating the reasons for matching.

g. Theoretical questions

- Author the solution in own words.
- Provide a clear idea regarding the concept being discussed.
- Explain the key terms and technical words clearly.
- Low level explanation is desirable.
- Provide diagrams, if necessary.

h. Problematic Questions:

- Identify and explain the given constraints/inputs of the problem.
- In each step give detailed explanation what is going to be achieved/calculated.
- If formula is required in a step; write it, give description of the formula, and its significance in the step.
- Don't provide direct answers.

i. Pseudo code and Flowcharts:

- Read and understand the question carefully and identify all the requirements.
- Follow the instructions given in the question and present the pseudo code, flowchart in the same order.
- Pseudo code style should match the style given in the textbook.
- Use Serif Draw Plus, Microsoft Paint, Microsoft Visio (or any other equivalent software) to draw flow-charts.
- Flow-chart symbols should be aligned neatly.

j. Case Study Questions:

- Read and understand the given case study.
- Read the questions given for the respective case study.
- Author the solution such that the ideas/suggestions/context discussed in the case study should reflect in the solution.
- Solution should be within the context of the given case study and if required within the context of the respective chapter.

Back