CS 159 Credit Exam

An increasing number of students entering the First Year Engineering program at Purdue University are bringing with them previous programming experience and a many of these students are interested in demonstrating that their experience is sufficient to earn credit for CS 159.

What is the format of the exam?

Fifty multiple-choice questions will be selected from a large pool of problems that cover the material introduced in CS 159. The exam will be taken on Blackboard, be limited to two hours, and the only acceptable resources for the exam are an ASCII table and an operator precedence table. Students are expected to be at a level of mathematics that includes preparation for calculus. No calculators are permitted and most of the expressions to evaluate will involve addition, subtraction, multiplication, division, and modulus.

Questions on the exam will ask students to interpret code and understand the fundamental terminology of the C programming language. There is no code writing on the exam.

What is required to pass the exam?

The credit exam is similar in content to a final exam offered in CS 159. The average on the final exam in a typical semester is around 70% and this will be considered the threshold necessary to establish credit by exam.

What advice do you have for students who have previously programmed in another language like JAVA or C++?

The credit exam is primarily a C language exam. The differences between programming languages at times can be minor but the details matter and this is what the exam will evaluate. A student who successfully completes CS 159 on campus will understand these details and those attempting to establish credit by exam will be held to the same standard.

Future courses that you may take in Engineering will assume mastery of the material in CS 159. A generic previous programming experience may be insufficient for success in future courses.

Why is my transfer credit not evaluated as equivalent to CS 159?

The interpretation of equivalent has been taken to mean an identical set of topics and languages. Most courses taken at other institutions fail to meet this mark or a level of rigor that is expected from an experience at Purdue University. Passing this credit exam will demonstrate that the experience and knowledge retained is sufficient to establish credit in CS 159.

What other considerations should a student take into account prior to attempting the credit exam?

There is only ONE opportunity to earn the passing score. It is important to emphasize that future classes will assume an equivalent experience to CS 159. There is no benefit to moving into a higher level course only to withdraw or earn a poor grade because the previous experience was insufficient.

A set of problems similar to those asked on the exam have been provided for your review.

- The answers are included at the end of the exam document.
What specific material is covered in CS 159?

  - Chapter 1 – Introduction to Computers
  - Chapter 2 – Introduction to the C Language
  - Chapter 3 – Structure of a C Program
  - Chapter 4 – Functions
  - Chapter 5 – Selection – Making Decisions
  - Chapter 6 – Repetition
  - Chapter 8 – Arrays
    - Includes a knowledge of the searching and sorting algorithms as introduced in the text.
  - Chapter 9 – Pointers
    - Section 9.1 only
  - Chapter 10 – Pointer Applications
    - Sections 10.1 – 10.4 (malloc)
  - Chapter 11 – Strings
    - Sections 11.1 – 11.5

Is there any MATLAB or Octave on the credit exam?

- Yes, you should be familiar with external file functions including the standard `fopen`, `fclose`, `fscanf`, `feof`. Only a small number of questions (1-3) related to these file functions will appear on the credit exam.
- A good resource for this material can be found on-line:
  - [https://www.gnu.org/software/octave/octave.pdf](https://www.gnu.org/software/octave/octave.pdf)
CS 159 Test-Out Practice Exam

1. Which of the following statements regarding constants is FALSE?
   A. When evaluated in an expression both variable and constant values have data types.
   B. A literal constant is an unnamed constant used to specify data.
   C. The use of symbolic/defined constants is encouraged to give meaning to operands in an expression.
   D. None of the above.

2. Which of the following statements regarding the selection of a data type for a variable is FALSE?
   A. The operations that can be performed on a value is limited by its type.
   B. The amount of memory necessary to store a value depends on its type.
   C. How a value is stored in the memory of the computer depends on its type.
   D. None of the above.

Use the program below for problems 3 – 4

```c
#include<stdio.h>
define SHORT_PI 3.14

int main()
{
    double diameter = 2.1;
    double circumference;
    double area;
    int precision;
    int width;

    circumference = SHORT_PI * diameter;
    area = circumference * diameter;
    width = 21 - circumference;
    precision = (area - (int) area) * 5;

    printf("-=-=-=-=-=-=-=-=-=-=-=-=-=-\n");
    printf("Circumference: %12.*lf\n", precision, circumference);
    printf("-=-=-=-=-=-=-=-=-=-=-=-=-=-\n");
    printf("Area: %*lf\n", width, area);
    return(0);
}
```

3. Which of the following is the output generated by the first two print statements in the program above?
   A. -=-=-=-=-=-=-=-=-=-=-=-=-=-
      Circumference: 6.5940
   B. -=-=-=-=-=-=-=-=-=-=-=-=-=-
      Circumference: 6.594
   C. -=-=-=-=-=-=-=-=-=-=-=-=-=-
      Circumference: 6.5940
   D. -=-=-=-=-=-=-=-=-=-=-=-=-=-
      Circumference: 6.594

4. Which of the following is the output generated by the last two print statements in the program above?
   A. -=-=-=-=-=-=-=-=-=-=-=-=-=-
      Area: 13.847400
   B. -=-=-=-=-=-=-=-=-=-=-=-=-=-
      Area: 13.8474
   C. -=-=-=-=-=-=-=-=-=-=-=-=-=-
      Area: 13.847400
   D. -=-=-=-=-=-=-=-=-=-=-=-=-=-
      Area: 13.8474
Use the code segment below for problems 5 – 8

```c
int x = 7;
int y = 10;
int z = 5;
int result = 0;

result = ++y - 10 || z - 5 && x++;
result += y++ - 11 || z++ - 5 && x++;
result += y + 1 > 11 && (z++ >= 6 || x++);

printf("result: %d\n", result);
printf("x: %d\n", x);
printf("y: %d\n", y);
printf("z: %d\n", z);
```

5. Which of the following is the first line of output generated by the code segment above?
   A. result: 0
   B. result: 1
   C. result: 2
   D. None of the above.

6. Which of the following is the second line of output generated by the code segment above?
   A. x: 7
   B. x: 8
   C. x: 9
   D. None of the above.

7. Which of the following is the third line of output generated by the code segment above?
   A. y: 11
   B. y: 12
   C. y: 13
   D. None of the above.

8. Which of the following is the fourth line of output generated by the code segment above?
   A. z: 5
   B. z: 6
   C. z: 7
   D. None of the above.

9. Which of the following is NOT a complement of the logical expression below?
   ```c
   x == y || x != z
   ```
   A. !(x != y && x == z)
   B. x != y && x == z
   C. !(x == y || x != z)
   D. None of the above.
10. Which of the following would be an invalid user-defined function name?
   A. _123
   B. printf_
   C. 1stcall
   D. None of the above

11. Which of the following statements regarding user-defined functions is FALSE?
   A. A variable declared in the local declaration section of a function can have the same identifier as one of the parameters within the same function.
   B. Data sent from the calling function to the function being called will be received in the same order in which it was passed.
   C. Parameters are defined as local variables in the first line of the function definition and should not be re-defined within the local declaration section of the function.
   D. None of the above.

12. Which of the following statements regarding the short-circuit method of evaluating logical expressions is TRUE?
   A. The truth value of an expression will change depending on whether a short-circuit method occurs or not.
   B. The short-circuit method is limited to use in selection statements.
   C. The short-circuit method is limited to use in repetition statements.
   D. None of the above.

13. Which of the following statements regarding if constructs is FALSE?
   A. Proper indentation of else will determine to which if it belongs in a nested construct.
   B. Nested if constructs can be used to test different variables.
   C. While there is no limit to the number of levels of nesting in a nested if construct, a larger number of levels may make the code difficult to read.
   D. None of the above.

14. Which of the following statements regarding the conditional expression is TRUE?
   A. Only a single terminal semicolon is used to terminate a conditional expression.
   B. A conditional expression cannot have another conditional expression as one of its executable actions.
   C. The logical expression of a conditional expression is limited to operands of the integer and character data types.
   D. None of the above.

15. Which of the following statements regarding the rules of the switch construct is TRUE?
   A. Two case labels can have the same constant expression value.
   B. No two case labels can be associated with the same set of actions.
   C. The control expression that follows the keyword switch must be a numeric type.
   D. None of the above.
#include<stdio.h>

int numDaysInMonth(int);

int main()
{
    int i;
    int num = 0;

    for(i = 1; i < 6; i++)
    {
        num += numDaysInMonth(i);
    }

    printf("num: %d\n", num);
    return(0);
}

int numDaysInMonth(int month)
{
    int days = 0;

    switch(month % 2)
    {
        case 1:  days = 31;
        break;
        case 0:  days = month == 2 ? 28 : 30;
    }

    return(days);
}

16. Which of the following is the output generated by the print statement in the program above?
   A. num: 151
   B. num: 152
   C. num: 154
   D. None of the above.

17. Which of the following is would be the output generated by the print statement in the program above if the control expression of the switch were changed from (month % 2) to (month % 2 == 0) ?
   A. num: 151
   B. num: 152
   C. num: 154
   D. None of the above.
int div;
int x = 30;
int total = 0;

while(x <= 41)
{
    div = 2;
    while(x % div != 0 && div <= sqrt(x))
    {
        div++;
    }
    if(div > sqrt(x))
    {
        total += div;
    }
    x++;
}

printf("total: %d\n", total);
printf("div: %d\n", div);

18. Which of the following is the output generated by the first print statement in the code segment above?
   A. total: 20
   B. total: 21
   C. total: 25
   D. None of the above.

19. Which of the following is the output generated by the second print statement in the code segment above?
   A. div: 2
   B. div: 6
   C. div: 7
   D. None of the above.
Use the code segment below for problems 20 – 21

```c
int a;
int b;
int c;
int d = 0;

for(a = 64; a > 1; a /= 2)
{
    for(b = 1; b < 27; b *= 3)
    {
        for(c = 2; c < 10; c += 2)
        {
            d++;
        }
    }
}
printf("sum: %d\n", a + b + c);
printf("d: %d\n", d);
```

20. Which of the following is the output generated by the first print statement in the code segment above?
   A. sum: 40
   B. sum: 38
   C. sum: 36
   D. None of the above.

21. Which of the following is the output generated by the second print statement in the code segment above?
   A. d: 84
   B. d: 66
   C. d: 60
   D. None of the above.
Use the code segment below for problems 22 – 23

```c
int findValues(int);

int main()
{
    int val;

    val = findValues(154);

    printf("val: %d\n", val);
    return(0);
}

int findValues(int n)
{
    int seek = 0;

    if(n > 0)
    {
        seek = findValues(n / 5);
        if(n % 2 == 0 && n % 10 > seek)
        {
            seek = n % 10;
        }
    }
    return(seek);
}
```

22. Which of the following is the output generated by the print statement in the code segment above?
   A. val: 4
   B. val: 1
   C. val: 6
   D. None of the above.

23. What is the total number of times that the findValues function is called in the code segment above?
   A. Four
   B. Five
   C. Six
   D. None of the above.

24. Which of the following statements regarding file functions is TRUE?
   A. The fscanf function requires the name of the external data file as one of its parameters.
   B. The fopen function requires the name of the external data file as its only parameter.
   C. The fclose function requires the name of the external data file as its only parameter.
   D. None of the above.
Use the program below for problems 25 – 27

```c
#include<stdio.h>
#define SIZE 8

int searchArray(int[], int);

int main()
{
    int x[SIZE] = {9, 5, 15, 3, 14, 17, 7};
    int y[SIZE] = {11, 15, 14, 17, 16, 0, 5};
    int i;
    int ct = 0;

    for(i = 0; i < SIZE; i++)
    {
        ct += searchArray(y, x[i]);
    }

    printf("ct: %d\n", ct);
    printf("y[0]: %d\n", y[0]);
    printf("y[5]: %d\n", y[5]);
    return(0);
}

int searchArray(int arry[], int n)
{
    int i;
    int result = 0;

    for(i = 0; i < SIZE; i++)
    {
        if(arry[i] == n)
        {
            result = 1;
            arry[i] = -1;
            i = SIZE;
        }
    }

    return(result);
}
```

25. Which of the following is the output generated by the first print statement in the program above?
   A. ct: 0    B. ct: 4    C. ct: 5    D. None of the above.

26. Which of the following is the output generated by the second print statement in the program above?
   A. y[0]: 0    B. y[0]: -1    C. y[0]: 11    D. None of the above.

27. Which of the following is the output generated by the third print statement in the program above?
   A. y[5]: 0    B. y[5]: -1    C. y[5]: 16    D. None of the above.
Use the program below for problems 28 – 30

```c
#include<stdio.h>

void changeArray(int[], int);

int main()
{
    int x[8] = {2, 3, 5, 4, 1, 0, 7, 6};
    int i;

    for(i = 0; i < 8; i += 2)
    {
        changeArray(x, i);
    }

    printf("x[0] = %d\n", x[0]);
    printf("x[3] = %d\n", x[3]);
    printf("x[7] = %d\n", x[7]);

    return(0);
}

void changeArray(int y[], int i)
{
    int j;

    for(j = 1; j < 8 - i; j++)
    {
        y[j] += y[j - 1];
    }
}

28. Which of the following is the output generated by the first print statement in the program above?
   A. x[0] = 8
   B. x[0] = 5
   C. x[0] = 2
   D. None of the above.

29. Which of the following is the output generated by the second print statement in the program above?
   A. x[3] = 26
   B. x[3] = 31
   C. x[3] = 57
   D. None of the above.

30. Which of the following is the output generated by the third print statement in the program above?
   A. x[7] = 28
   B. x[7] = 29
   C. x[7] = 30
   D. None of the above.
Use the program below for problems 31 – 33

#include<stdio.h>
#define SIZE 9

int main()
{
    int i;
    int x[SIZE] = {13, 11, 22, 32, 15, 23, 28, 19, 18};

    for(i = 0; i < SIZE; i++)
    {
        x[i] -= x[SIZE - i - 1];
        x[SIZE - i - 1] += x[i];
        x[i] = x[SIZE - i - 1] - x[i];
    }

    printf("x[2] = %d\n", x[2]);
    printf("x[4] = %d\n", x[4]);
    printf("x[8] = %d\n", x[8]);
    return(0);
}

31. Which of the following is the output generated by the first print statement in the program above?
   A. x[2] = 22
   B. x[2] = 28
   C. x[2] = 44
   D. None of the above.

32. Which of the following is the output generated by the second print statement in the program above?
   A. x[4] = 0
   B. x[4] = 15
   C. x[4] = 30
   D. None of the above.

33. Which of the following is the output generated by the third print statement in the program above?
   A. x[8] = 13
   B. x[8] = 18
   C. x[8] = 26
   D. None of the above.

34. When passing a multi-dimensional array to a function where would you NOT include the sizes of those dimensions beyond the first?
   A. In the declaration of the function being called.
   B. In the definition of the function being called.
   C. In the call of the function.
   D. None of the above.
Use the code segment below for problem 35

```c
int x[5] = {6, 9, 3, 0, 4};
int y[5] = {0};

y = x;
printf("y[0] = %d\n", y[0]);
```

35. Which of the following describes the integer value generated by the print statement in the code segment above?
A. The value displayed will be the integer 6.
B. The value displayed will be the memory address represented by the array x.
C. No integer value will be displayed due to a compiler error regarding the assignment statement.
D. None of the above.

Use the program below for problems 36 – 38

```c
#include<stdio.h>
#include<string.h>

int main()
{
    char str1[21] = "ABC Company";
    char str2[31] = "ABC Corp";
    int result;

    result = strcmp(str1, str2);
    printf("result #1: %d\n", result);

    result = strlen(str2);
    printf("result #2: %d %d\n", result, str2[result]);

    strcpy(str1, str2);
    printf("str1: %s str2: %s\n", str1, str2);

    return(0);
}
```

36. Which of the following is the output generated by the first print statement in the program above?
A. result #1: 5
B. result #1: 0
C. result #1: -5
D. None of the above.

37. Which of the following is the output generated by the second print statement in the program above?
A. result #2: 8 0
B. result #2: 8 112
C. result #2: 7 112
D. None of the above.

38. Which of the following is the output generated by the third print statement in the program above?
A. str1: ABC Corp str2: ABC Corp
B. str1: ABC Company str2: ABC Company
C. str1: ABC Corpany str2: ABC Corp
D. None of the above.
39. Which of the following statements regarding arrays and user-defined functions is FALSE?
   A. It is possible to pass multiple individual elements of an array to a function by value.
   B. It is possible to pass individual elements of an array to a function by address.
   C. It is possible to pass a whole array to a function by value.
   D. None of the above.

   Use the code segment below for problem 40

   ```c
   int x = 3;
   int *y;
   int *z;

   y = &x;
   z = y;
   (*z)--;

   printf("result: %d\n", *y + *z);
   ```

40. Which of the following is the output generated by the print statement in the code segment above?
   A. result: 2
   B. result: 3
   C. result: 4
   D. result: 6
<table>
<thead>
<tr>
<th>Char</th>
<th>Dec</th>
<th>Char</th>
<th>Dec</th>
<th>Char</th>
<th>Dec</th>
<th>Char</th>
<th>Dec</th>
</tr>
</thead>
<tbody>
<tr>
<td>delimiter</td>
<td>0</td>
<td>space</td>
<td>32</td>
<td>@</td>
<td>64</td>
<td>`</td>
<td>96</td>
</tr>
<tr>
<td>(soh)</td>
<td>1</td>
<td>!</td>
<td>33</td>
<td>A</td>
<td>65</td>
<td>a</td>
<td>97</td>
</tr>
<tr>
<td>(stx)</td>
<td>2</td>
<td>&quot;</td>
<td>34</td>
<td>B</td>
<td>66</td>
<td>b</td>
<td>98</td>
</tr>
<tr>
<td>(etx)</td>
<td>3</td>
<td>#</td>
<td>35</td>
<td>C</td>
<td>67</td>
<td>c</td>
<td>99</td>
</tr>
<tr>
<td>(eot)</td>
<td>4</td>
<td>$</td>
<td>36</td>
<td>D</td>
<td>68</td>
<td>d</td>
<td>100</td>
</tr>
<tr>
<td>(enq)</td>
<td>5</td>
<td>%</td>
<td>37</td>
<td>E</td>
<td>69</td>
<td>e</td>
<td>101</td>
</tr>
<tr>
<td>(ack)</td>
<td>6</td>
<td>&amp;</td>
<td>38</td>
<td>F</td>
<td>70</td>
<td>f</td>
<td>102</td>
</tr>
<tr>
<td>(bel)</td>
<td>7</td>
<td>'</td>
<td>39</td>
<td>G</td>
<td>71</td>
<td>g</td>
<td>103</td>
</tr>
<tr>
<td>(bs)</td>
<td>8</td>
<td>(</td>
<td>40</td>
<td>H</td>
<td>72</td>
<td>h</td>
<td>104</td>
</tr>
<tr>
<td>(ht)</td>
<td>9</td>
<td>)</td>
<td>41</td>
<td>I</td>
<td>73</td>
<td>i</td>
<td>105</td>
</tr>
<tr>
<td>(nl)</td>
<td>10</td>
<td>*</td>
<td>42</td>
<td>J</td>
<td>74</td>
<td>j</td>
<td>106</td>
</tr>
<tr>
<td>(vt)</td>
<td>11</td>
<td>+</td>
<td>43</td>
<td>K</td>
<td>75</td>
<td>k</td>
<td>107</td>
</tr>
<tr>
<td>(np)</td>
<td>12</td>
<td>,</td>
<td>44</td>
<td>L</td>
<td>76</td>
<td>l</td>
<td>108</td>
</tr>
<tr>
<td>(cr)</td>
<td>13</td>
<td>-</td>
<td>45</td>
<td>M</td>
<td>77</td>
<td>m</td>
<td>109</td>
</tr>
<tr>
<td>(so)</td>
<td>14</td>
<td>.</td>
<td>46</td>
<td>N</td>
<td>78</td>
<td>n</td>
<td>110</td>
</tr>
<tr>
<td>(si)</td>
<td>15</td>
<td>/</td>
<td>47</td>
<td>O</td>
<td>79</td>
<td>o</td>
<td>111</td>
</tr>
<tr>
<td>(dle)</td>
<td>16</td>
<td>0</td>
<td>48</td>
<td>P</td>
<td>80</td>
<td>p</td>
<td>112</td>
</tr>
<tr>
<td>(dc1)</td>
<td>17</td>
<td>1</td>
<td>49</td>
<td>Q</td>
<td>81</td>
<td>q</td>
<td>113</td>
</tr>
<tr>
<td>(dc2)</td>
<td>18</td>
<td>2</td>
<td>50</td>
<td>R</td>
<td>82</td>
<td>r</td>
<td>114</td>
</tr>
<tr>
<td>(dc3)</td>
<td>19</td>
<td>3</td>
<td>51</td>
<td>S</td>
<td>83</td>
<td>s</td>
<td>115</td>
</tr>
<tr>
<td>(dc4)</td>
<td>20</td>
<td>4</td>
<td>52</td>
<td>T</td>
<td>84</td>
<td>t</td>
<td>116</td>
</tr>
<tr>
<td>(nak)</td>
<td>21</td>
<td>5</td>
<td>53</td>
<td>U</td>
<td>85</td>
<td>u</td>
<td>117</td>
</tr>
<tr>
<td>(syn)</td>
<td>22</td>
<td>6</td>
<td>54</td>
<td>V</td>
<td>86</td>
<td>v</td>
<td>118</td>
</tr>
<tr>
<td>(etb)</td>
<td>23</td>
<td>7</td>
<td>55</td>
<td>W</td>
<td>87</td>
<td>w</td>
<td>119</td>
</tr>
<tr>
<td>(can)</td>
<td>24</td>
<td>8</td>
<td>56</td>
<td>X</td>
<td>88</td>
<td>x</td>
<td>120</td>
</tr>
<tr>
<td>(em)</td>
<td>25</td>
<td>9</td>
<td>57</td>
<td>Y</td>
<td>89</td>
<td>y</td>
<td>121</td>
</tr>
<tr>
<td>(sub)</td>
<td>26</td>
<td>:</td>
<td>58</td>
<td>Z</td>
<td>90</td>
<td>z</td>
<td>122</td>
</tr>
<tr>
<td>(esc)</td>
<td>27</td>
<td>;</td>
<td>59</td>
<td>[</td>
<td>91</td>
<td>{</td>
<td>123</td>
</tr>
<tr>
<td>(fs)</td>
<td>28</td>
<td>&lt;</td>
<td>60</td>
<td>\</td>
<td>92</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(gs)</td>
<td>29</td>
<td>=</td>
<td>61</td>
<td>]</td>
<td>93</td>
<td>}</td>
<td>125</td>
</tr>
<tr>
<td>(rs)</td>
<td>30</td>
<td>&gt;</td>
<td>62</td>
<td>^</td>
<td>94</td>
<td>~</td>
<td>126</td>
</tr>
<tr>
<td>(us)</td>
<td>31</td>
<td>?</td>
<td>63</td>
<td>_</td>
<td>95</td>
<td>(del)</td>
<td>127</td>
</tr>
</tbody>
</table>
This page lists C operators in order of *precedence* (highest to lowest). Their *associativity* indicates in what order operators of equal precedence in an expression are applied.

<table>
<thead>
<tr>
<th>Operator</th>
<th>Description</th>
<th>Associativity</th>
</tr>
</thead>
<tbody>
<tr>
<td>()</td>
<td>Parentheses (function call)</td>
<td>left-to-right</td>
</tr>
<tr>
<td>[]</td>
<td>Brackets (array subscript)</td>
<td></td>
</tr>
<tr>
<td>++ --</td>
<td>Postfix increment/decrement</td>
<td></td>
</tr>
<tr>
<td>++ --</td>
<td>Prefix increment/decrement</td>
<td>right-to-left</td>
</tr>
<tr>
<td>+ -</td>
<td>Unary plus/minus</td>
<td></td>
</tr>
<tr>
<td>!</td>
<td>Logical negation</td>
<td></td>
</tr>
<tr>
<td>(type)</td>
<td>Cast (change type)</td>
<td></td>
</tr>
<tr>
<td>*</td>
<td>Dereference</td>
<td></td>
</tr>
<tr>
<td>&amp;</td>
<td>Address</td>
<td></td>
</tr>
<tr>
<td>sizeof</td>
<td>Determine size in bytes</td>
<td></td>
</tr>
<tr>
<td>* / %</td>
<td>Multiplication/division/modulus</td>
<td>left-to-right</td>
</tr>
<tr>
<td>+ -</td>
<td>Addition/subtraction</td>
<td>left-to-right</td>
</tr>
<tr>
<td>&lt; &lt;=</td>
<td>Relational less than/less than or equal to</td>
<td>left-to-right</td>
</tr>
<tr>
<td>&gt; &gt;=</td>
<td>Relational greater than/greater than or equal to</td>
<td>left-to-right</td>
</tr>
<tr>
<td>== !=</td>
<td>Relational is equal to/is not equal to</td>
<td>left-to-right</td>
</tr>
<tr>
<td>&amp;&amp;</td>
<td>Logical AND</td>
<td>left-to-right</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>?:</td>
<td>Ternary conditional</td>
<td>right-to-left</td>
</tr>
<tr>
<td>=</td>
<td>Assignment</td>
<td>right-to-left</td>
</tr>
<tr>
<td>+= -=</td>
<td>Addition/subtraction assignment</td>
<td>right-to-left</td>
</tr>
<tr>
<td>*= /=</td>
<td>Multiplication/division assignment</td>
<td>left-to-right</td>
</tr>
<tr>
<td>%=</td>
<td>Modulus assignment</td>
<td></td>
</tr>
<tr>
<td>,</td>
<td>Comma (separate expressions)</td>
<td>left-to-right</td>
</tr>
</tbody>
</table>
Sample Exam Answers