Chapter 5
Style Requirements

Matters of Style

Programming style can greatly enhance the readability of a program or it can detract from it and make a program confusing, difficult to read, debug and later modify. Style elements include the position of internal documentation, capitalization of identifiers, indentation and spacing.

Spacing and indentation add to the visual organization of a program where capitalization gives a visual clue to a programmer as to what an identifier represents. We will use the following style requirements in CS 1A and CS 1B.

YOU ARE RESPONSIBLE FOR LEARNING THIS STYLE AND USING IT IN ALL PROGRAMS TURNED IN. REFER TO THIS SECTION OFTEN.

Basic Style Requirements
Computer Science Department
Saddleback College

1. Named constants will be in all caps.

   TAX_RATE PROGRAMMER

2. Variables begin with a lower case letter and only the first letter of each successive word will be capitalized.

   hoursWorked payRate grossPay

3. The declaration section shall contain one identifier per line along with the data type.

```
int firstNum;          int firstNum,
int secondNum;             secondNum,
int thirdNum;              thirdNum;
```

   CORRECT **INCORRECT**

4. The declaration section must contain a data table stating the use of the variable or named constant and how its value is obtained/used.

```
int firstNum;     // first value to average - INPUT
int secondNum;    // second value to average - INPUT
int thirdNum;     // third value to average – INPUT
float average;    // average of three values – CALC & OUTPUT
```

   CORRECT

```
int firstNum; // first value to average - INPUT
int secondNum; // second value to average - INPUT
int thirdNum; // third value to average – INPUT
float average; // average of three values – CALC & OUTPUT
```

   INCORRECT

```
int firstNum;     // input value
int secondNum;    // input value
int thirdNum;     // input value
float average;    // calculated average
```

   INCORRECT

5. Proper indentation is required – SET THE TAB STOP TO 2 IN THE CODEWARRIOR EDITOR.

```
{
    cout << "hello" << endl;
}
```
CORRECT

```
{
cout << "hello" << endl;
}
```
INCORRECT

```
{
        cout << "hello" << endl;
}
```
INCORRECT

```
{
  cout << "hello" << endl;
}
```
INCORRECT

6. Document the executable code by section on the line ABOVE the code it references. The sections of the program are separated with blank lines.

```
#include <iostream.h>

void main(void)
{
    int num1;          // first value to average – INPUT
    int num2;          // second value to average – INPUT
    float average;     // average of two values – CALC & OUTPUT

    // get numbers to average from user
    cout << "Enter first value to average: ";
    cin >> num1;
    cout << " Enter second value to average: ";
    cin >> num2;

    // calculate the average
    average = float(num1 + num2) / 2;

    // output the average
    cout << "\n\nThe average of the two numbers is " << average;
}
```

CORRECT

```
#include <iostream.h>
void main(void)
{
    int num1;          // first value to average – INPUT
    int num2;          // second value to average – INPUT
    float average;     // average of two values – CALC & OUTPUT
    // get numbers to average from user
    cout << "Enter first value to average: ";
    cin >> num1;
    cout << " Enter second value to average: ";
    cin >> num2;
    // calculate the average
    average = float(num1 + num2) / 2;
    // output the average
    cout << "\n\nThe average of the two numbers is " << average;
}
```

INCORRECT

```
#include <iostream.h>
void main(void)
{
    int num1;          // first value to average – INPUT
    int num2;          // second value to average – INPUT
    float average;     // average of two values – CALC & OUTPUT

    cout << "Enter first value to average: ";      // read number from user
    cin >> num1;
    cout << " Enter second value to average: "; // read number from user
    cin >> num2;

    average = float(num1 + num2) / 2;             // calculate the average

    cout << "\n\nThe average of the two numbers is " << average;  // output the average
}
```

INCORRECT

7. Your program must begin with a description of what the program does.

```
// This program calculates and outputs the average of two numbers.  The two numbers are obtained
// from the user of the program.

#include <iostream.h>
```

CORRECT

```
// Number average program
#include <iostream.h>
```

INCORRECT

8. Your program must contain your name, class section, title of homework assignment and due date.

```
#include <iostream.h>

void main(void)
{
   // declaration section here

   cout << "Joe Programmer\n";
   cout << "CS 1B MW 10:30 – Noon\n";
   cout << "Page 151 - #\4\n";;
   cout << "Due: 1/19/2000\n\n";


   // documentation for next section here
}
```

CORRECT

```
#include <iostream.h>
void main(void)
{
   // declaration section here

   cout << "Joe Programmer\n";
   cout << "CS 1B Homework Assignment\n\n";

   // documentation for next section here
}
```

INCORRECT

9. DO NOT INITIALIZE VARIABLES IN THE DECLARATION SECTION. Initialize variables just before their use in the program.

```
int count;

count = 0;
```

CORRECT

```
int count = 0;
```

INCORRECT

10. A block format shall be used for if and if/else statements. French braces shall be used even for single statements.

```
if (hours > 40)
{
    cout << "Overtime pay is due";
}
```

CORRECT

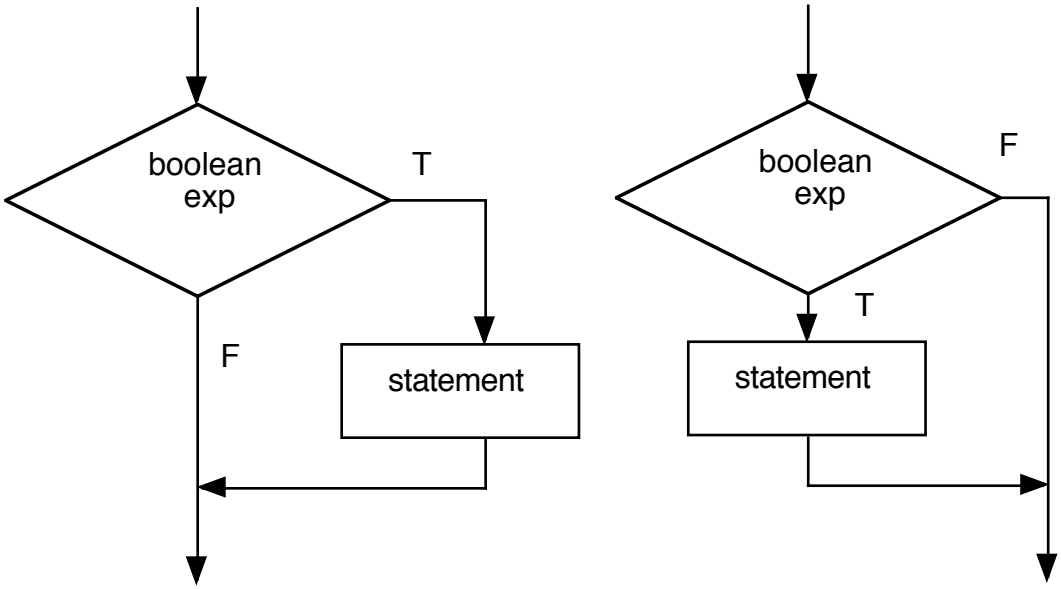
```
if (hours > 40) cout << "Overtime pay is due";
```

INCORRECT

```
if (hours > 40)
    cout << "Overtime pay is due";
```

INCORRECT

FLOWCHART STYLE (if statement)



CORRECT **INCORRECT**

11. Use the block form of the if/else statement. The block format for the if/else requires the else to be placed on a line by itself. Note the indentation. French braces line up with their corresponding if and else and the code contained in the braces is indented using a tab stop of 2.

```
if (hours > 40)
{
    cout << “Overtime is due”;
}
else
{
    cout << “No overtime is due”;
}
```

CORRECT

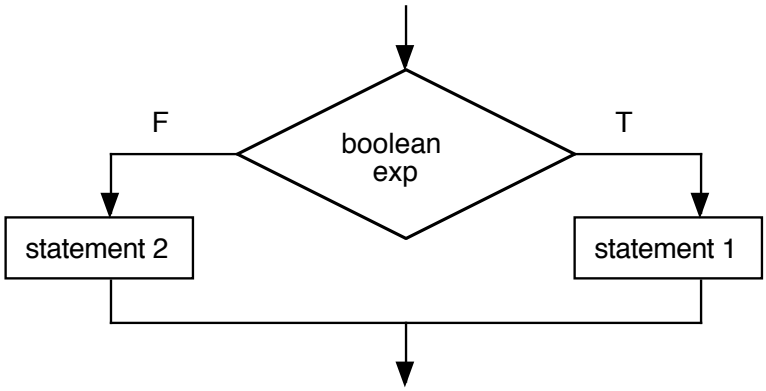
```
if(hours > 40)
{
cout << "Overtime is due";
}
else
{
cout << "No overtime is due";
}
```

INCORRECT

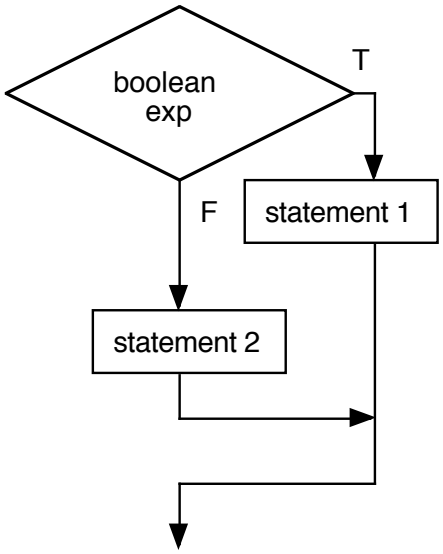
```
if(hours > 40)
    {
    cout << "Overtime is due";
    }
    else
{
cout << "No overtime is due";
}
```

INCORRECT

FLOWCHART STYLE (if/else statement)



CORRECT



INCORRECT

12. Nested if statements shall also use the block format.

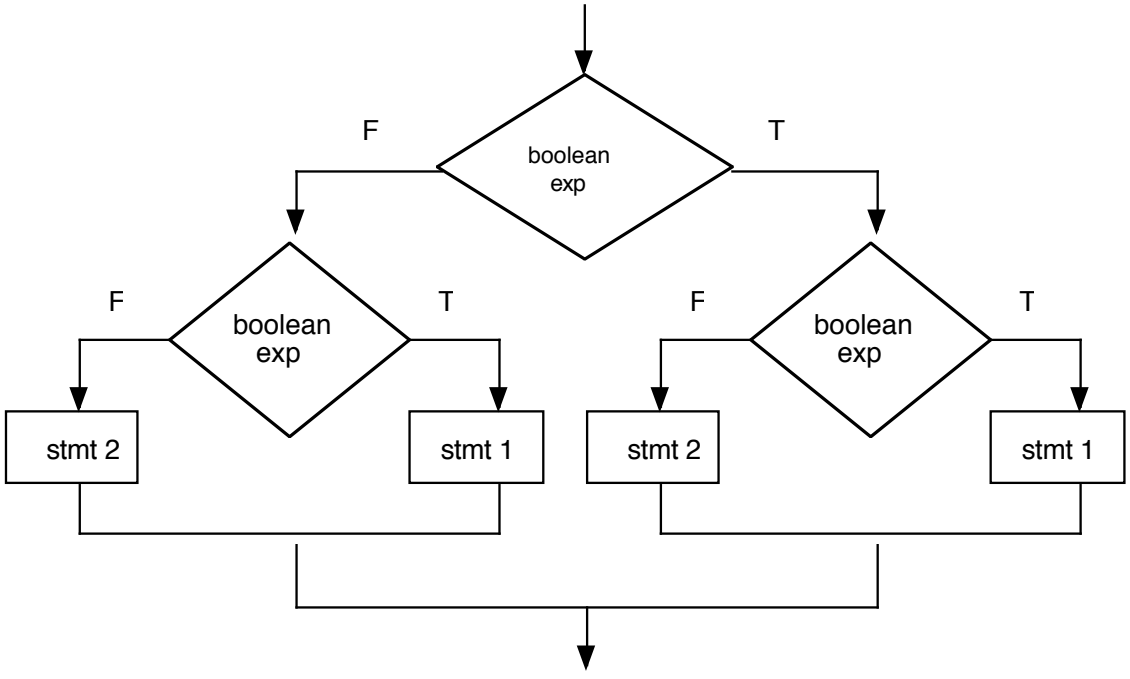
```
if (hours == 40)
{
    cout << "Full time";
}
else
{
    if (hours < 40)
    {
        cout << "Part time";
    }
    else
    {
        cout << "Overtime due";
    }
}
```

CORRECT

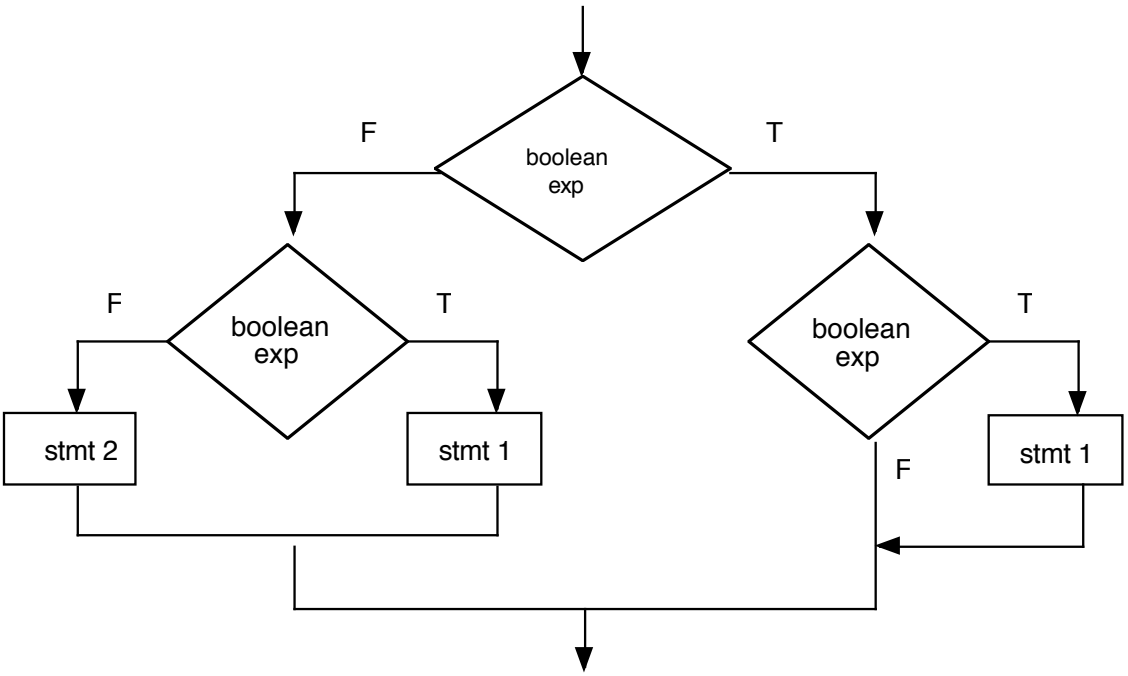
```
if (hours == 40)
{
        cout << "Full time";
}
else if (hours < 40)
{
    cout << "Part time";
}
    else
{
    cout << "Overtime due";
}
```

INCORRECT

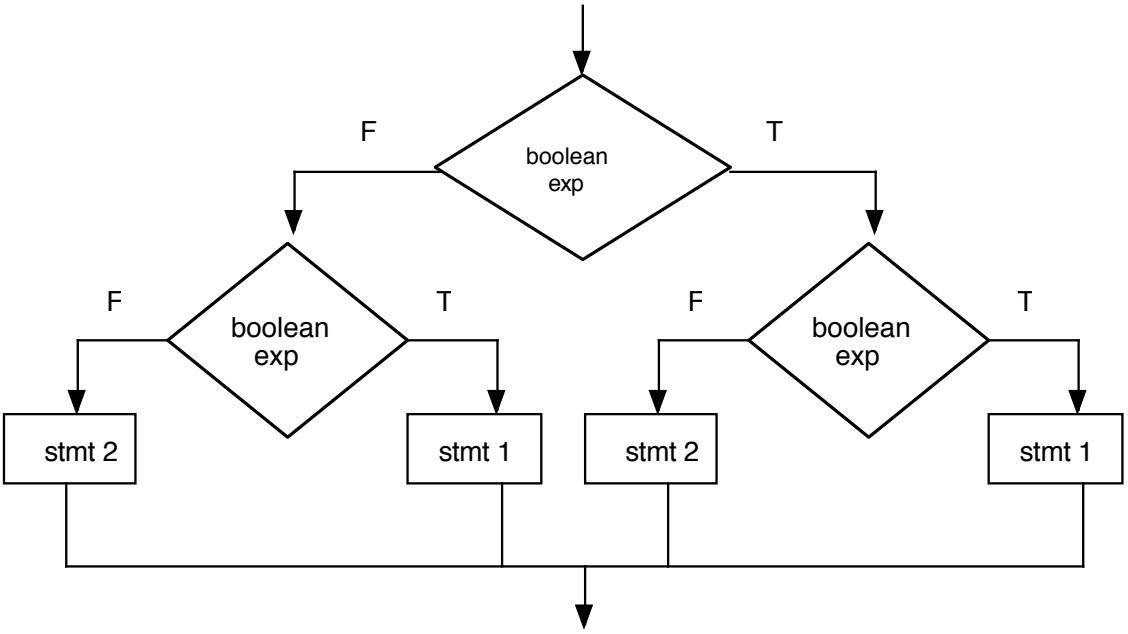
FLOWCHARTING STYLE (nested if)



CORRECT



CORRECT



INCORRECT

13. The switch statement shall be indented as follows.

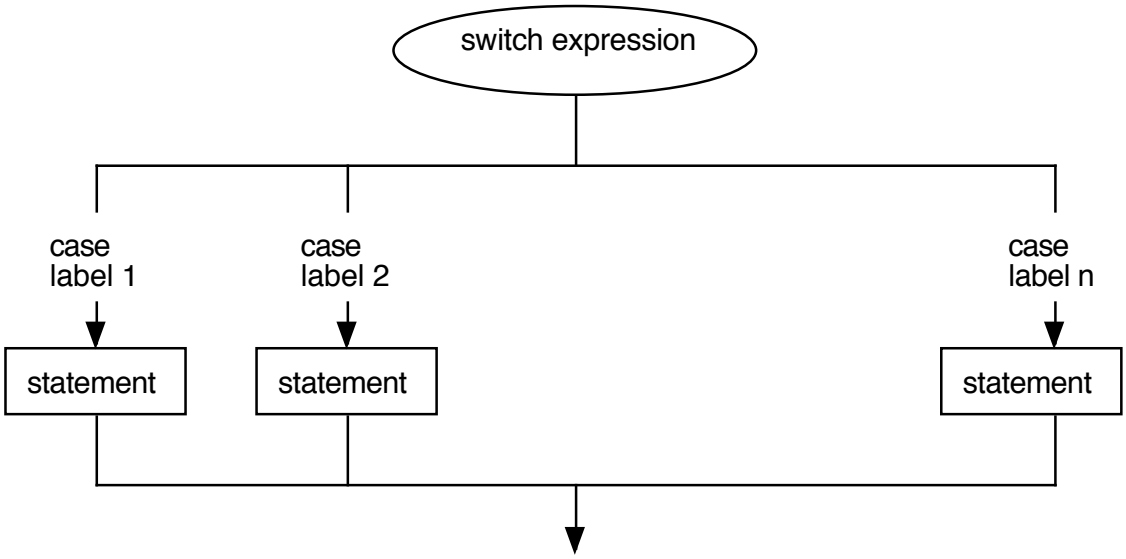
```
switch (maritalStatus)
{
    case 'M'  :  cout << "Married";
                 break;
    case 'S'  :  cout << "Single";
                 break;
    case 'D'  :  cout << "Divorced";
                 cout << "Choose wisely next time";
                 break;
    default   :  cout << "Invalid code";
                 break;
}
```

CORRECT

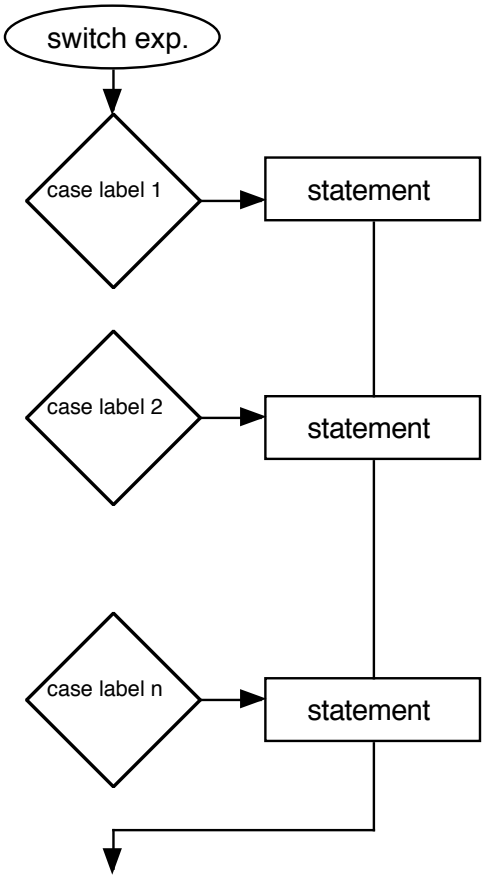
```
switch (maritalStatus)
{
case 'M'  :  cout << "Married";
             break;
case 'S'  :  cout << "Single";
break;
case 'D'  :  cout << "Divorced";
             cout << "Choose wisely next time";
             break;
default      :  cout << "Invalid code";
     break;
}
```

INCORRECT

FLOWCHART STYLE (switch statement)



CORRECT



INCORRECT

14. Format for the *while* loop.

```
cout << "Enter a score: ";
cin >> score;
while (score != -999)
{
    scoreTot += score;
    scoreCount ++;
    cout << "Enter a score: ";
    cin >> score;
}
```

CORRECT

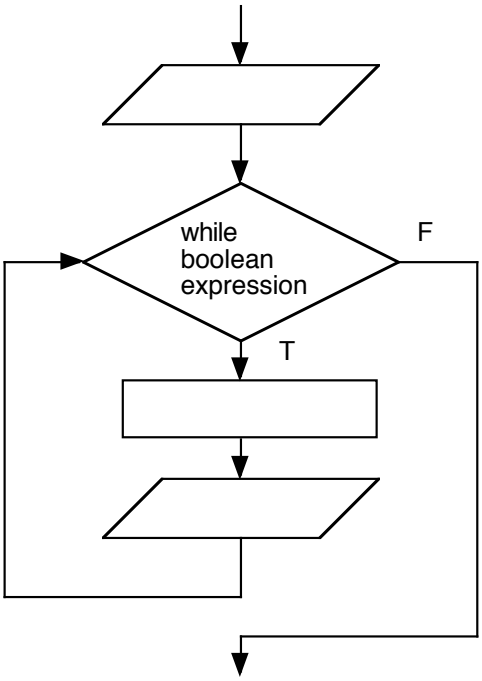
```
cout << "Enter a score: ";
    cin >> score;
    while (score != -999) {
    scoreTot += score;
    scoreCount ++;
    cout << "Enter a score: ";
    cin >> score; }
```

INCORRECT

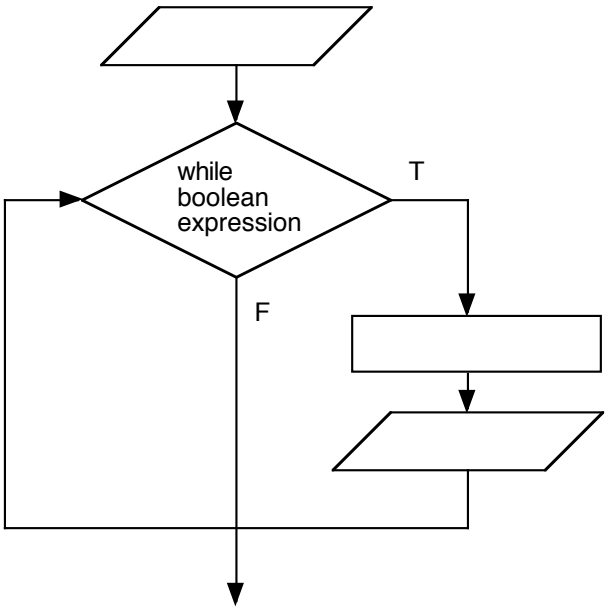
```
cout << "Enter a score: ";
cin >> score;
while (score != -999)
{
scoreTot += score;
scoreCount ++;
cout << "Enter a score: ";
cin >> score;
}
```

INCORRECT

FLOWCHART STYLE (while loop)



CORRECT



INCORRECT

15 Format for the *do .. while* loop

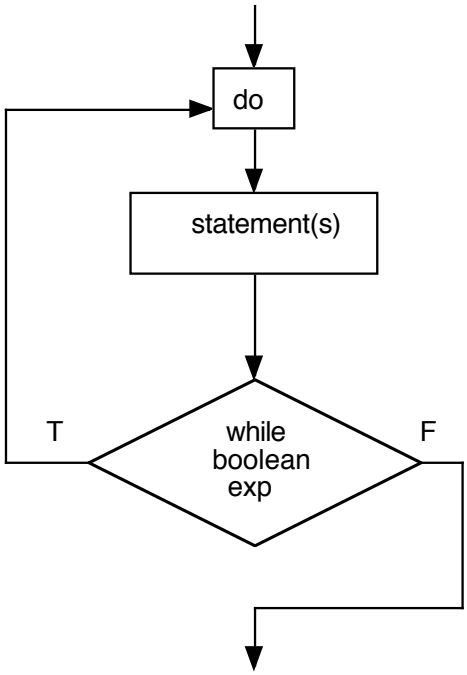
```
do
{
    cin >> val;
    tot = tot + val;
}
while (val != -999);
```

CORRECT

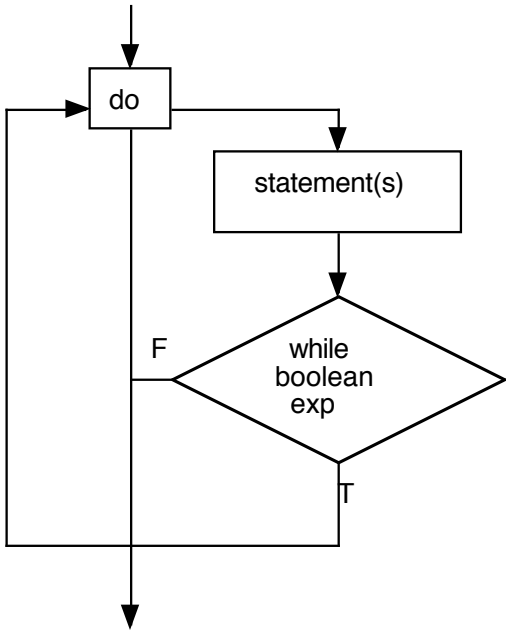
```
do {
    cin >> val;
    tot = tot + val;
    } while (val != -999);
```

INCORRECT

FLOWCHARTING STYLE (do..while loop)



CORRECT



INCORRECT

16. Format for the *for* loop

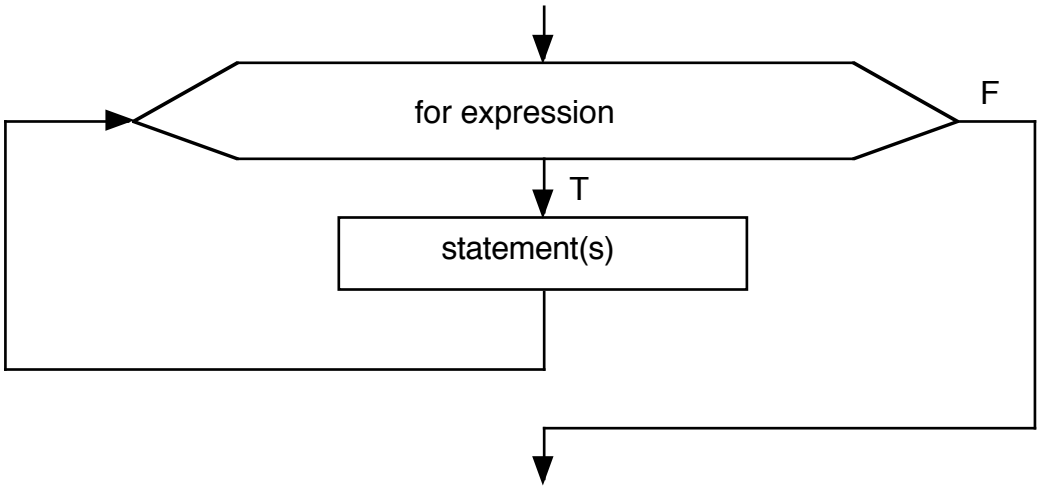
```
for (int i = 1; i <= 10; i++)
{
    cin >> val;
    tot = tot + val;
}
```

CORRECT

```
for (int i = 1; i <= 10; i++) {
cin >> val;
tot = tot + val; }
```

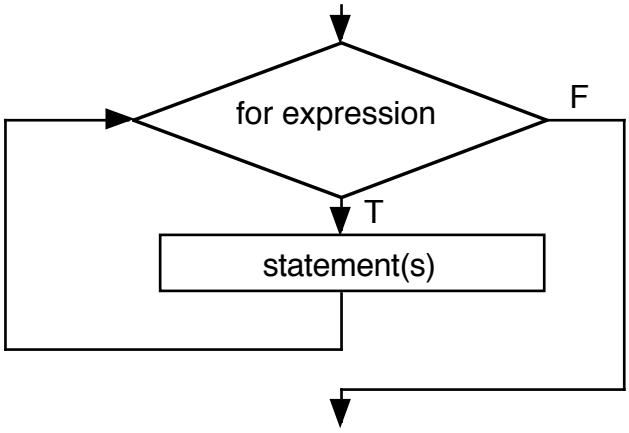
INCORRECT

FLOWCHARTING STYLE (for loop)



CORRECT

NOTE: The diamond is NOT used for a counted loop.



INCORRECT