

Letter grades

Marcus Birkenkrahe (pledged)

May 4, 2024

Contents

1	Problem	1
2	Submission	2
3	Tip: how to analyse a programming problem	2
3.1	Pseudocode	3
3.2	BPMN model	3
3.3	C code	3
4	Solution with switch and break statements	6
4.0.1	Pseudocode	6
4.0.2	BPMN model	8
4.0.3	C code	8
5	Checking division by integer	9

1 Problem

- Using the `switch` statement, write a program that converts a *numerical* grade into a *letter* grade.

- Example run:

```
Enter numerical grade: 84
Letter grade: B
```

- Use the following grading scale:

Numerical grade	Letter grade
90-100	A
80-89	B
70-79	C
60-69	D
0-59	F

- Print an error message if the grade is larger than 100 or less than 0.
- *Hint:* You can break the grade into two digits, then use a `switch` statement to test the ten's digit.

2 Submission

- Submit the code as an **Emacs Org mode file** in Canvas.
- The submission must include the `#+RESULTS:` and the usual header (`#+TITLE`, `#+AUTHOR` with (pledged)).
- Extra credit (5 pts) for submitting a BPMN diagram of the algorithm included. Send a screenshot of the diagram (as SVG file `bpmn.svg`) to my via email to get the points, and add it to your Org-mode file like this:

```
#+attr_html: :width 600px
[[[bpmn.svg]]]
```

3 Tip: how to analyse a programming problem

- A programming solution requires identifying
 1. problem (given in the text - is it clear?)
 2. constants (which values do not change?)
 3. variables (which values do change?)
 4. statements (what needs computing?)
- For any but trivial problems, spending time on gathering and structuring this information before beginning to code will save you lots of debugging time

- Constants: The letter grades are constants and do not change. However, they are not needed for computations like numerical constants (e.g. pi).
- Variables:
 - the numerical grade
- Statements:
 - printing the letter grade
 - EITHER cascading if...else for each letter grade/interval
 - OR cases for each letter grade/interval
- Helpful: pseudocode gives you the **story** of the algorithm
- Helpful: a diagrammatic model (BPMN) shows you the **logic** and helps uncover special cases, dangling ends etc.

~* Solution with cascading if statements

3.1 Pseudocode

You can start with pseudocode: the pseudocode for the if...else cascade could look like this:

```
if 90 <= grade <= 100 letter grade is A else if 80 <= grade <=
89 letter grade is B else if 70 <= grade <= 79 letter grade is C
else if 60 <= grade <= 69 letter grade is D else if 59 <= grade
<= 0 letter grade is F end if
```

3.2 BPMN model

You could create a BPMN model to illustrate the decision points. The new insight when creating the BPMN model is the possibility of not giving a grade if the number grade is not in [0,100].

3.3 C code

- In this case, the C code can almost be read off the pseudocode.
- Each if statement tests a conditional expression. We have to know how to test if the grade is in an interval.

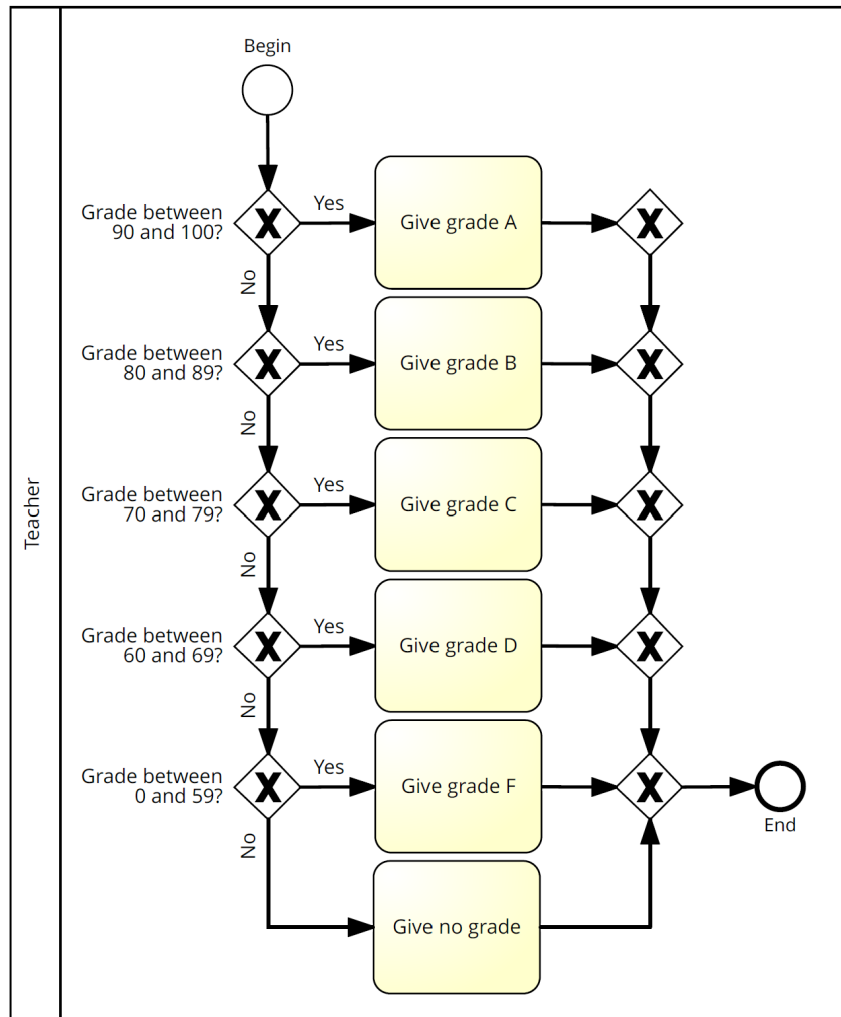


Figure 1: BPMN model of problem (if solution)

- For example, to test if a variable `i` is in the mathematical interval `[90,100]`, use the expression `i >= 90 && i <= 100`.
- We define test input by writing it to an input file `grade`:

```
echo 84 > grade_if_1
```

```
int i = 0;
scanf("%d", &i);

if (i >= 90 && i <= 100) {
    printf("Numerical grade %d\n", i);
    printf("Letter grade A\n");
} else if (i >= 80 && i <= 89) {
    printf("Numerical grade %d\n", i);
    printf("Letter grade B\n");
} else if (i >= 70 && i <= 79) {
    printf("Numerical grade %d\n", i);
    printf("Letter grade C\n");
} else if (i >= 60 && i <= 69) {
    printf("Numerical grade %d\n", i);
    printf("Letter grade D\n");
} else if (i <= 59 && i >= 0) {
    printf("Numerical grade %d\n", i);
    printf("Letter grade F\n");
} else if (i < 0 || i > 100) {
    printf("Invalid input %d\n", i);
}
```

- Let's take care of the possibility that the input lies outside of `[0,100]`. Values above 100 could e.g. be the result of extra credit. Values below 0 could be an input mistake.

In the program ??, values outside of `[0,100]` have no effect whatsoever - nothing is printed.

```
echo 84 > grade_if_2
```

```
int i = 0;
scanf("%d", &i);
```

```

if (i >= 90) { // removed the upper bound
    printf("Numerical grade %d\n", i);
    printf("Letter grade A\n");
}
else if (i >= 80 && i <= 89) {
    printf("Numerical grade %d\n", i);
    printf("Letter grade B\n");
}
else if (i >= 70 && i <= 79) {
    printf("Numerical grade %d\n", i);
    printf("Letter grade C\n");
}
else if (i >= 60 && i <= 69) {
    printf("Numerical grade %d\n", i);
    printf("Letter grade D\n");
}
else if (i <= 59 && i >= 0) {
    printf("Numerical grade %d\n", i);
    printf("Letter grade F\n");
}
else if (i < 0) { // include values below lower bound
    printf("Input %d not valid\n", i);
}

```

4 Solution with switch and break statements

4.0.1 Pseudocode

This pseudocode includes the possibility of a wrong entry (you didn't have to implement that).

```

if grade is not in [0,100] switch to A if grade is a multiple of 10
or 9 B if grade is a multiple of 8 C if grade is a multiple of 7 D
if grade is a multiple of 6 F if grade is below 59 end if

```

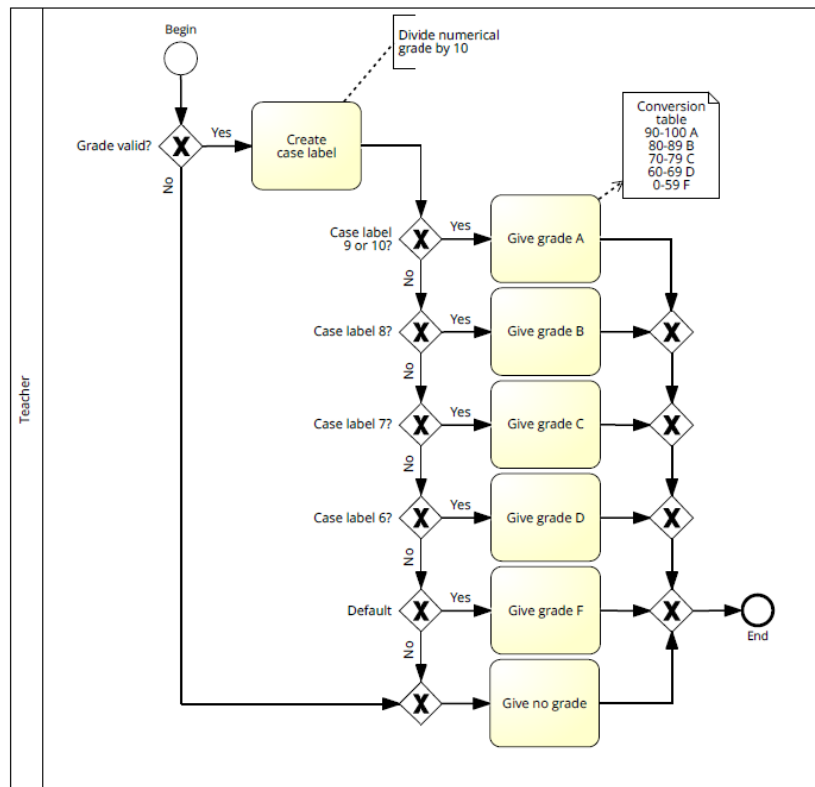


Figure 2: BPMN model of problem (switch solution)

4.0.2 BPMN model

4.0.3 C code

- Dividing the numerical grade by 10 gives five cases with the labels 10 to 6, corresponding to the letter grades A through D, and grades 0-59 as the default grade F.
- An `if` clause around the `switch` statement takes care of `score` entries that are outside of the range or that are just wrong like the entry 'A'.
- With so many statements, it is advisable to use brackets to identify the exact range of the statements.

```
echo 84 > grade_switch
```

```
int score;
```

```
printf("Enter score (0-100): \n");
```

```
scanf("%d", &score);
```

```
if ( score >= 0 && score <=100 ) {  
    switch ( score / 10 ) {  
        case 10 :      case 9 :  
            printf("Score %d means letter grade A", score);  
            break;  
        case 8 :  
            printf("Score %d means letter grade B", score);  
            break;  
        case 7 :  
            printf("Score %d means letter grade C", score);  
            break;  
        case 6 :  
            printf("Score %d means letter grade D", score);  
            break;  
        default :  
            printf("Score %d means letter grade F", score);  
            break;  
    }  
} else {  
    printf("Score %d is outside of the permitted range.\n", score);  
}
```


5 Checking division by integer

```
for (int i = 100; i >= 0; i--) {  
    printf("i = %d, i/10 = %d\n", i, i/10);  
}
```

```
int i = 0;  
printf("i = %d, i/10 = %d\n", i, i/10);
```