CS1010E: Programming Methodology

Tutorial 03: Selection

06 Feb 2017 - 10 Feb 2017

1. Discussion Questions

(a) [Bad Practices] What is/are the output of *badly written* code fragments below?

i.
```
int a = 3, b = 4, ans = 1;
if(b > a > 1) {
  ans = a + b;
} else {
  ans = a - b;
}
printf("%d", ans);
```
i. _____

ii.
```
int a = 3, b = 4, ans = 1;
if(a > 3)
  ans  = a + b; ans += 1;
printf("%d", ans);
```
ii. _____

iii.
```
int a = 3, b = 4, ans = 1;
if(a >= 3) ans = a + b; else ans = a - b;
printf("%d", ans);
```
iii. _____

iv.
```
int a = 3, b = 4, ans = 1;
if(a >= 3)
  if(b > 4)
    ans = a + b;
else
  ans = a - b;
printf("%d", ans);
```
iv. _____

v.
```
int a = 2, ans = 1;
if(a%2) ans = 0;
printf("%d %d", a, a%2);
```
v. _____

2. Program Analysis

(a) [Operation Precedence] What is/are the output of code fragments below?

i.
```
int x = 5, y = 8, z = 13, ans;
ans = x + y < z ? ++x < y ? x++ : y++ : x < z ? x++ : z++;
printf("%d %d %d %d", x, y, z, ans);
```
i. _____

ii.
```
int x = 0, ans = 0;
switch(x) {
  case 1:  ans = 1;   x++;
  case 0:  ans += 2;  ++x;
  case 2:  if(x == 0) break; ans++;
  case 3:  ans -= 4;  break;
  default: ans = 0
} printf("%d", ans);
```
ii. _____

iii.
```
int a = 1, b = -1, c = 0, d = -1, e = 0, f = 1, ans;
ans = --a || b++ && c-- && ++d || ++e || --f;
printf("%d %d %d %d %d %d %d", a, b, c, d, e, f, ans);
```
iii. _____

(b) [Abstraction] State (*in English*), what is the purpose of the following code fragments below?

i.
```
// given a, b, c as user input
if(a > b)
  if(a > c)
    ans = a;
  else
    ans = c;
else
  if(b > c)
    ans = b;
  else
    ans = c;
printf("%d", ans);
```
i. _____

ii.
```
// given x as user input
if(x < 0)
  ans = -x;
else
  ans = x;
printf("%d", ans);
```
ii. _____

(c) [Limit of Values] What is/are the output of code fragments below?

i.
```
if(0.7 == 0.3 + 0.4)
  printf("Equal");
else
  printf("Not Equal");
```
i. _____

3. Designing a Solution

(a) [Computation] Given that the 1^{st} of January, 2017 falls on a Sunday, determine the day of the week for any given date in 2016. Note that it is possible to be done without using any repetition construct (*which some of you may not have yet learned*). Write your program below:

```
#define SUN 0   #define MON 1   #define TUE 2   #define WED 3
#define THU 4   #define FRI 5   #define SAT 6
#define JAN 1   #define FEB 2   #define MAR 3   #define APR 4
#define MAY 5   #define JUN 6   #define JUL 7   #define AUG 8
#define SEP 9   #define OCT 10  #define NOV 11  #define DEC 12

int main() {
  int day, month, week = 0; scanf("%d %d", &day, &month);
  /* Compute Day of the Week */
  switch(week) {
    case SUN: printf("Sunday");     break;
    case MON: printf("Monday");     break;
    case TUE: printf("Tuesday");    break;
    case WED: printf("Wednesday");  break;
    case THU: printf("Thursday");   break;
    case FRI: printf("Friday");     break;
    case SAT: printf("Saturday");   break;
  }
  return 0;
}
```

(b) [Boolean Logic] The Gregorian calendar has a pretty complicated rule for determining leap year. The rule that says "year that is divisible by 4" is in fact, *incomplete*. The exception to this rule is that for years that are divisible by 100, it is a leap year only if it is also divisible by 400. Otherwise, it is considered a common year.

Given the description above, the rule for leap year can be summarized as follows for any given `year`:

1. `year` that is divisible by 4 is leap year, *except for*
2. `year` that is divisible by 100 –which is common year– *unless it is*
3. `year` that is divisible by 400 –which is leap year.

Write the code to determine if a given `year` is a leap year or not. You are given two sets of template below, answer for both cases:

i. Multiple checks, *using **only**[1] arithmetic and relational operations*:

```
int main() {
  int year; bool leap; scanf("%d", &year);
  if ( ____________________________ ) {
    leap = true;
  } else if ( ____________________________ ) {
    leap = false;
  } else if ( ____________________________ ) {
    leap = true;
  } else {
    ________________ ;
  }
  return 0;
}
```

ii. Single checks, *using arithmetic, relational, and **logical** operations*:

```
int main() {
  int year; bool leap; scanf("%d", &year);
  if ( ____________________________________________________________ ) {
    leap = true;
  } else {
    ________________ ;
  }
  return 0;
}
```

4. Challenge

(a) [Flowchart] Programming is not always about writing codes, but it is always about solving problems. Flowchart is a viable option for programming as well. Diagram 1 shows a simple program for patient diagnostic. The convention for the flowchart is given below:

- Oval: Starting point
- Diamond: Decision point, user input is given via standard input
- Square: Diagnosis reached, output should be printed on standard output

Every decision has two choices: `yes` or `no`. We will represent `no` with value `0` and `yes` with value `1`. Assume that inputs are given in the following order: `fever?`, `sore throat?`, `cough?`, `chills?`, `swelling?`, and `nausea?`. Write the program corresponding to the flowchart. Write your program below:

[1] Remember that arithmetic operations involve: [`+`, `-`, `/`, `%`, etc], relational operations involve: [`==`, `>=`, `<=`, `>`, `<`, etc], and logical operations involve: [`&&`, `||`, etc]

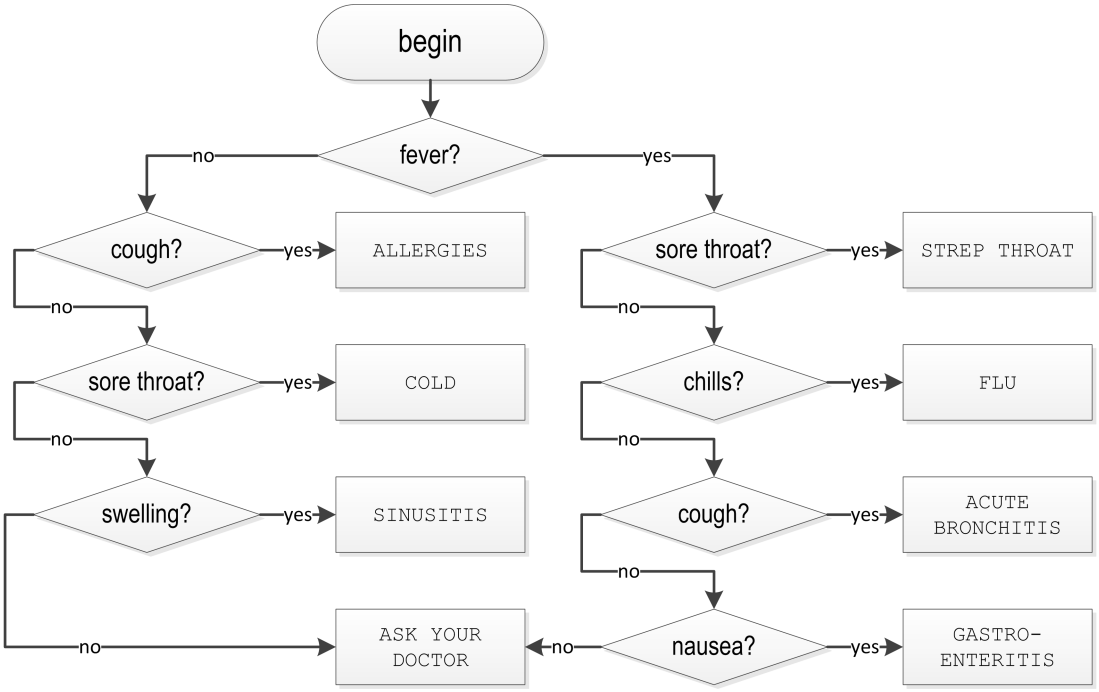

Diagram 1: A simple flowchart for patient diagnostic.

```
int main() {
  bool fever, sore, cough, chills, swelling, nausea
  /* Flowchart Program */
  return 0;
}
```