Introduction to Computer Science
Tutorial 2: variables, arithmetic operators, printf(), scanf()

1. Linux Processes

Exercise 1. Using CTRL+z, `bg`, `fg` and CTRL+c, do the following: get 3 `sleep` processes running in a terminal at the same time and show that they are running with the command **`jobs`**), then terminate all 3 of them. Choose the sufficient amounts of seconds for the `sleep` commands to last enough time (e.g., more than 30 seconds). You should see the output of `jobs` showing the three `sleep` processes as "running".

2. C programs with scanf() and printf()

Exercise 2. Write a C program that asks the user for 3 integer values, then prints their average (using the + and / operators).

Exercise 3. Write a C program that asks the user for 2 integer values, that represent the length and the width of rectangle, then prints the perimeter and area of the rectangle.

3. C programs with conditions

Exercise 4. Write a C program that does the following:

- declare two int variables *a* and *b*
- ask the user two values for these variables *a* and *b*
- then, if *a* is equal to b, print "*a* and *b* are equal"
- else, if *a* is strictly greater than *b*, print "*a* is greater"
- else print "*b* is greater"
- finally, print the absolute value of the difference between *a* and *b*

Hint: if x is an arithmetic expression, the expression `abs(x)` gives its absolute value.

3. Manual execution

"Manual execution" works as follows. A C program is given (typically without boilerplate like #include <...>, main(){...}), and you have to create a table that shows the execution of the program. Each row starts with the statement being evaluated. Each variable has its own column, and indicates new values when updated. There is a “condition” column that indicates TRUE/FALSE if the statement is a condition.

The point of the exercise is not only to correctly perform the assignments and arithmetic operations, but also to correctly follow the executio of the program with respect to conditional structures (if, if-else).

For example:

```
int a = 1;
if (a < 4)
    a = a + 4;
else
    a = a – 1;
if (a > 4)
    a = a + 1;
else
    a = a – 1;
```

	a		condition
`int a=1`	1		
`a < 4`			true
`a = a + 4`	5		
`a > 4`			true
`a = a + 1`	6		

Exercise 5. For each one of the following C programs, write the table of manual execution. The table should contain a "statement" column, then a column for every variable in the program, then a "condition" column.

```
/* 1.1 */
int a = 35;
int b = 7;
a = a % 10;
b = a + b;
if (a + b > 10) {
    a = 0;
}
else {
    b = 0;
}
```

```
/* 1.2 */
int i = 2;
if (i > 0) {
    i = i + 2;
}
if (i > 3) {
    i = i – 3;
}
```

```c
/* 1.3 */
int b = 7;
int c = 0;
if (b > 9) {
    c = 1;
}
else if (b > 5) {
    c = 2;
}
else {
    c = 3;
}
```

4. Simplifying Programs with "if-else" statements

Exercise 6. Simplify the following programs by using else statements whenever relevant (you can also download them from Moodle).

```c
// badif01.c
main() {
    int x;
    scanf("%d", &x);
    if (x > 0)
        printf("x is positive");
    if (x < 0)
        printf("x is negative");
    if (x == 0)
        printf("x is zero");
}
```

```c
// badif02.c
main(){
    int bill_y, wallet_y, wallet_usd;
    int d_to_y = 7;

    scanf("%d", &bill_y);
    scanf("%d", &wallet_y);
    scanf("%d", &wallet_usd);

    if (wallet_y >= bill_y)
        printf("Pay in yuans.");
    if (wallet_y < bill_y) {
        if (wallet_usd * d_to_y >= bill_y)
            printf("Pay in dollars.");
        if (wallet_usd * d_to_y >= bill_y)
            printf("Cannot pay.");
    }
}
```