Project 5

Problem 1 – Multiple of 11

Checking if a number X is a multiple of another number Y is easy when it is small enough to be stored in a variable of integer type (e.g., int, long int). We just have to check if the remainder of the division of X by Y is zero (i.e., X%Y == 0). However, when the number is too large, this can be a tricky task. Luckily, when dealing with multiples of 11 (Y=11), there is another way of performing this check: if the difference of the sum of the digits at odd places and even places of X is 0 or a multiple of 11, then X is a multiple of 11 as well.

Example #1:

X = 323455693

3	2	3	4	5	5	6	9	3

(3+3+5+6+3) - (2+4+5+9) = 20 - 20 = 0

The difference is 0, thus X is a multiple of 11

Example #2:

X = 31416

3	1	4	1	6

(3+4+6) - (1+1) = 13 - 2 = 11

The difference is 11, thus X is a multiple of 11

Considering that X can be a very large number (up to 1000 digits), can you write a program that identifies multiples of 11?

Hint #1: Read the input number as a string, otherwise you will not be able to handle so many digits. Hint #2: The sum of digits in odd places and even places will be small numbers and can be stored in **int** variables.

Hint #3: To convert a digit stored in a variable named **ch** to an integer value in the range [0-9], you can use the following equation: **(ch-'0')**

Example #1:

```
Enter the value of X: 323455693
323455693 is a multiple of 11
```

Example #2:

```
Enter the value of X: 31416
```

```
31416 is a multiple of 11
```

Example #3:

```
Enter the value of X: 7
7 is not a multiple of 11
```

Example #4:

```
Enter the value of X:
19536628991284076702291442896531030074228 28
19536628991284076702291442896531030074228 28 is a multiple of 11
```

Problem 2 – Command-line Arguments

```
./a.out all
Output: In order
```

We define the command-line argument to be in order if the characters of the input

1) are alphabetic letters, lower case or upper case.
2) any two neighboring letters (regardless of case) are in order, for example, ‘c’ and ‘k’ are in order but ‘s’ and ‘b’ is not in order because ‘c’ is less than ‘k’ and ‘s’ is greater than ‘b’, considering their ASCII values.
3) if two neighboring letters are same, they are considered in order.

The program determines if the command-line argument is in order.

1) Assume the command-line argument contains two or more characters.
2) Convert the characters to lower case before comparison.
3) Character handling functions are allowed.
4) The program should include the following function:

```
int in_order(char *word);
```

The function expects word to point to a string containing the string to be checked if it’s in order. The function returns 1 if word is in order, and 0 otherwise.

5) The program should also check if the correct number of arguments are entered on the command line. There should be only one argument besides `./a.out`. If an incorrect number of arguments are entered, the program should display a message and exit.

6) The main function displays the output.

Example #1:

```
./a.out all
Output: In order
```

Example #2:

```
./a.out littlepigs
Output: Not in order
```

Example #3:

```
./a.out cS
Output: In order
```

Example #4:

```
./a.out F28
Output: Not in order
```

1. Program names:
   project5_multiple.c
   project5_in_order.c

2. Compile.

gcc –Wall project5_multiple.c
gcc –Wall project5_in_order.c

3. Change Unix file permission on Unix:

chmod 600 project5_multiple.c
chmod 600 project5_in_order.c

4. Test your program with the shell script

chmod +x try_project5_multiple
./try_project5_multiple

chmod +x try_project5_in_order
./try_project5_in_order