2WF90: Number Theory programming assignment

Hans Sterk, Benne de Weger

September 2017

Contents

1 Number Theory programming assignment

1.1 Introduction

The programming assignment for the first part of the course consists of writing code ('from scratch') for arithmetic with large integers, also modular arithmetic. This is mainly meant as preparation for the second programming assignment on finite fields, which is described in a separate document.

1.2 Role in assessment

This programming assignment contributes 5% to your 2WF90 grade.

1.3 General guidelines for the programming assignments

- The assignment is carried out preferably in triples or quadruples. The work should be your own, but, of course, you may use results from the lecture notes and lectures (include proper references).
- Include (short) comments in the code explaining what happens at various places.

- Describe in a separate text document (to be handed in in pdf format) what your approach is: how you represent various objects, how users should input polynomials, etc. Show, by including, for example, screenshots of your program at work, that your set-up works. The document contains
  - A title page with your name(s), student id('s), Project name ('Assignment 2WF90, Number Theory, September 2017');
  - Table of contents, division in sections.
  - Where relevant references to literature and the lecture notes. List of references at the end of the document.
  - A description of what your code can do and of the limitations of your code.
  - Illustrative examples.
  - Make sure that every group member has contributed a fair share of the assignment; state each member's contribution.

  There will be a place in Canvas to submit your work.

- Hand in the code including explanations how to install and run things on a Windows platform. It is *vital* that the process of installing and running your code works smoothly, requires no special knowledge, and is not time consuming. You may be asked to demonstrate your program or help with the installation.
- You cannot expect us to debug your code.

You will be graded on the following basis (this will account for maximally 5% of the 20% reserved for the programming assignment:

- **Maximal contribution:** 50%**: Working code**
  Working code for each of the objects and forms of arithmetic mentioned below. Contains comment lines so that readers can find their way easily.
- **Maximal contribution:** 50%**: Description of the code**
  - 10% : Your text document is well-structured and includes a title page with title (project name), student name(s), student id's, date; table of contents (and so sections); references at appropriate places to the lecture notes (or other literature). List of references.
  - 40%: Description, in correct English, is mathematically sound, with references to the lecture notes (and, if necessary, other literature). Your text explains your approach and the way a user can use the code. Provide examples of questions that one can answer given the code. Indicate limitations of your work.

2 Programming assignment on integer arithmetic

Write code 'from scratch' for basic arithmetic with large integers using representations with as radix a configurable integer $b \leq 16$). 'From scratch' means that you are not allowed to use large-integer libraries built into mainstream computer languages such as Java or Python, but your code should use from such a language only elementary arithmetic operations on small integers of at most 32 bits. Your code should be able to

- deal with the standard radix b representation of integers of large word length (at least several hundreds);
- inputs and outputs of arithmetic operations should be in this standard format;
- deal with positive and negative integers, and with zero;
- read input from a file with a specific format (an example file is provided);
- write output to a file (no format specified);
- the following arithmetic operations should be supported:
  - addition, subtraction;
  - multiplication by the primary school method;
  - multiplication by the method of Karatsuba (only dividing up long numbers in two parts, with recursion);
- for the multiplication operations, the software should keep track of the number of elementary additions/subtractions it does, and the number of elementary multiplications, and to be able to show those operation counts in the output.

Present some examples of different word sizes, and compare elementary operation counts of primary school and Karatsuba methods. Draw a conclusion based on the theory.

Test values are given in the example input file `example.txt`.

Appendix

Here is the contents of the input file `example.txt` for the Integer Arithmetic assignment. The file itself is provided on Canvas.

```
# Example input file your program should be able to process
#
# Explanation:
#
# empty line:         to be ignored by the program
#
# line starts with
# #:                  comment, to be ignored by the program
# [add]:              apply addition to the following two numbers
# [subtract]:         apply subtraction to the following two numbers
# [multiply]:         apply primary school multiplication to the following two numbers
# [karatsuba]:        apply Karatsuba multiplication to the following two numbers
# [x]:                first number to which the operation is to be applied
# [y]:                second number to which the operation is to be applied
# # [answer]:         you may ignore this line in your program;
#                     those lines with answers were added for your convenience;
#                     you may use them (e.g. without the hash) to have your program
#                     test its correct working
#
# the [multiply] and [karatsuba] examples below use the same values for [x] and [y]
# (for the same [radix]); the answers your program produces should be equal too

[radix] 2
[add]
[x] 11001100010110111100111010111001000000010101011111010011110101011111001101100011
00010100001011111010
[y] 10010110101101000101111001011100110010100010100010010101001110010100110001010011
10000101000001010111
# [answer] 101100011000100000010110100010101110010111000000011010010000111100111111
1011011010011001001101010001

[radix] 2
[subtract]
[x] 11110010011110001001000001010100110111111111011001101001011101011011001001100010
10011101011010001011
[y] 10010110101001101110101010001011100100101110111001001111010100010110001101101001
11011000110001011111
# [answer] 1011011110100011010010111001001010011010000100000011010001001000100111011
11100011000100101000101100

[radix] 2
```

```
[multiply]
[x] 11110011100110110111010011000101011110000110101001000011010111011111011011100100
01111010001010110011
[y] 111011000000011101000001011000101001010101100011011000110111010001010010010010001
00001111010100100110
# [answer] 11100000101000010001111010010110111111110011100000101011110110010010000110
000001001101111110010010011111100000001001010010001110101100101100010110100110001011000
10000010010001000111000001110111010110010010

[radix] 2
[karatsuba]
[x] 11110011100110110111010011000101011110000110101001000011010111011111011011100100
01111010001010110011
[y] 111011000000011101000001011000101001010101100011011000110111010001010010010010001
00001111010100100110
# [answer] 11100000101000010001111010010110111111110011100000101011110110010010000110
000001001101111110010010011111100000001001010010001110101100101100010110100110001011000
10000010010001000111000001110111010110010010

[radix] 3
[add]
[x] 1021002222220100111012120000201102021221
[y] 1012200010211212111211101000010010220000
# [answer] 2110210010202020000000221000211120011221

[radix] 3
[subtract]
[x] 2021112002201010101201001221200211210001
[y] 1211222222202101122110112222212210012122
# [answer] 102112002221201202020111221211001120102

[radix] 3
[multiply]
[x] 2010220211120222202210012012101022110011
[y] -2102021102022210001211210202210112011002
# [answer] -120120002010100221001101101212112211222120010101012101122110100122011021
11111022

[radix] 3
[karatsuba]
[x] 2010220211120222202210012012101022110011
[y] -2102021102022210001211210202210112011002
# [answer] -120120002010100221001101101212112211222120010101012101122110100122011021
11111022
```

```
[radix] 16
[add]
[x] df9e76d113895821c567
[y] 6a20b188675ab39e17a2
# [answer] 149bf28597ae40bbfdd09

[radix] 16
[subtract]
[x] 36d2b5154ab14bfabbf2
[y] f9d1495cfafe396ae4b1
# [answer] -c2fe9447b04ced7028bf

[radix] 16
[multiply]
[x] -eed50d6aa53e51691add
[y] -f9027b863f654daae6a8
# [answer] e84f8af471ab1bb45d20f1a95313171b2ade2f08

[radix] 16
[karatsuba]
[x] -eed50d6aa53e51691add
[y] -f9027b863f654daae6a8
# [answer] e84f8af471ab1bb45d20f1a95313171b2ade2f08
```