Petunjuk:

1. Buku yang digunakan: Gilberg, R. F. & Forouzan, B. A. (2005). **Data Structures- A Pseudocode Approach with C** (2nd ed.). Thomson Learning, Inc.
2. Problem yang akan dikerjakan: **Practice Sets** (halaman 139)
   - Exercises No. 1 – 8
   - Problems No. 10, 16, 17, 19, 22.
3. Lakukan MOD 5 terhadap digit terakhir pada NIM Anda.
4. Sisanya menentukan problem mana saja yang harus Anda kerjakan.
   - Sisa 0 : mengerjakan Exercises No. 1, 3a, 4a, 5b, 7c, and Problems No. 10
   - Sisa 1 : mengerjakan Exercises No. 2, 3b, 4b, 6a, 7d, and Problems No. 16
   - Sisa 2 : mengerjakan Exercises No. 1, 3c, 4c, 6b, 8a, and Problems No. 17
   - Sisa 3 : mengerjakan Exercises No. 2, 3d, 4d, 7a, 8b, and Problems No. 19
   - Sisa 4 : mengerjakan Exercises No. 1, 3a, 5a, 7a, 7b, and Problems No. 22

Exercise 1

Imagine we have two empty stacks of integers, s1 and s2. Draw a picture of each stack after the following operations:

```
push( s1, 3 )
push( s1, 5 )
push( s1, 7 )
push( s1, 9 )
push( s1, 11 )
push( s1, 13 )
while not isEmpty( s1 ) do
  pop( s1, x )
  push( s2, x )
endwhile
```

Exercise 2

Imagine we have two empty stacks of integers, s1 and s2. Draw a picture of each stack after the following operations:

```
push( s1, 3 )
push( s1, 5 )
push( s1, 7 )
push( s1, 9 )
push( s1, 11 )
push( s1, 13 )
while not isEmpty( s1 ) do
  pop( s1, x )
  pop( s1, x )
  push( s2, x )
endwhile
```

D
-
B
+
C

0
-
-
+
+
0

D
D
DB
DB-
DB-C
DB-C+

C
+
B
-
D

0
+
+
-
-

C
C
CB
CB+
CB+D
CB+D-
-D+BC

Exercise 3

Using manual transformation, write the following infix expressions in their **postfix** and **prefix** forms:

a. D – B + C
b. A * B + C * D
c. (A + B) * C – D * F + C
d. (A – 2 * (B + C) – D * E) * F

Exercise 4

Using manual transformation, change the following postfix or prefix expressions to **infix**:

a. A B * C – D +
b. A B C + * D –
c. + – * A B C D
d. – * A + B C D

Exercise 5

If the values of A, B, C, and D are 2, 3, 4, and 5, respectively, manually calculate the value of the following **postfix** expressions:

a. A B * C – D +
b. A B C + * D –

2 3 * 4 - 5 +
6 4 - 5 +
2 5 +
7

Exercise 6

If the values of A, B, C, and D are 2, 3, 4, and 5, respectively, manually calculate the value of the following **prefix** expressions:

a. + – * A B C D
b. – * A + B C D

Exercise 7

Change the following **infix** expressions **to postfix** expressions using a **stack**:

a. D – B + C
b. A * B + C * D
c. (A + B) * C – D * F + C
d. (A – 2 * (B + C) – D * E) * F

D
D
DB
DB-
DB-C
DB-C+

A
*
B
+
C
*
D

0 A
* A
* AB
+ AB*
+ AB*C
+* AB*C
+* AB*CD+*

AB*CD+*

Exercise 8

Determine the value of the following **postfix** expressions when the variables have the following values: A is 2, B is 3, C is 4, and D is 5.

a. A B C D * – +
b. D C * B A + –

Problem 10

Using stack, write the pseudocode for an algorithm that prints a decimal number as an **octal number**.

Note: similar problem can be found in Program 3-16 page 107: Convert Decimal to Binary

Problem 16

One of the applications of a stack is to **backtrack**—that is, to retrace its steps. As an example, imagine we want to read a list of items, and each time we **read a negative number** we must **backtrack** and **print the five numbers that come before the negative number** and then **discard the negative number**.

Use a stack to solve this problem.

- **Read** the numbers and **push** them into the stack (without printing them) until a negative number is read.
- At this time, **stop** reading and **pop five** items from the stack and print them.
- If there are **fewer than five items** in the stack, **print an error** message and **stop** the program.
- After printing the five items, **resume** reading data and placing them in the stack.
- When the **end of the file** is detected, **print** a message and the items **remaining** in the stack.
- Test your pseudocode with the following data:
  `1 2 3 4 5` **`-1`** `1 2 3 4 5 6 7 8 9 10` **`-2`** `11 12` **`-3`** `1 2 3 4 5`

Problem 17

Write the pseudocode for an algorithm called **copyStack** that copies the contents of one stack into another. The algorithm passes **two stacks**, the source stack and the destination stack. The **order** of the stacks must be **identical**.

Hint: Use a temporary stack to preserve the order.

Problem 19

A **palindrome** is a string that can be read backward and forward with the same result. For example, the following is a palindrome: `Able was I ere I saw Elba`.

Write a function to test if a string is a palindrome **using a stack**.

- You can **push** characters in the stack one by one.
- When you reach the **end of the string**, you can **pop** the characters and form a new string.
- If the two strings are **exactly the same**, the string is a palindrome.
- Note that palindromes **ignore spacing**, **punctuation**, and **capitalization**.

Test your function with the following test cases:

```
Go dog Madam, I'm Adam
Madam, I'm not a palindrome
```

Problem 22

Write a function to **check** whether the contents of two stacks are **identical**. Neither stack should be changed. You need to write a function that **prints** the contents of a stack to verify that your function works.