Tarea 3 - Programación y algoritmos
Giovanni Gamaliel López Padilla

Problema 1

Programa que verifique si un número de entrada dado es palíndomo. Un palíndromo, es un número o string que se lee igual en un sentido que en otro (por ejemplo: Ana, 121...). Usar una función que regrese un numero invertido (al revés).
El algoritmo que se creo es el siguiente:

```
1    reverse = 0
2    number = 12345
3    while number != 0:
4        aux = number % 10
5        reverse = reverse * 10 + aux
6        number = number / 10
```

Al final la variable reverse contendra el numero escrito al reves, esto es porque la operación de la linea 4 se obtiene el número que se encuentra en la posicion de las unidades, por ende es sumada en la linea 4. Al ser number una variable de tipo entero entonces al momento de realizar la operación de la linea 5 el numero de la parte decimal es eliminada, por ende en la proxima iteración en la linea 3 se tomara el que se encontraba en la posicion de las centenas. La función que realiza el algoiritmo es llamada reverse y se encuentra en el archivo reverse.h. El programa se encuentra en la carpeta Problema_1. Para compilar se uso el siguiente comando:

```
1    gcc -Wall -Wextra -Werror -pedantic -ansi -o main.out main.c -std=c11
```

El output esperado para un número palindromo es el siguiente:

```
1
2    Escribe el numero que quieres saber si es palindromo:
3    123454321
4
5
6    El numero 123454321 si es palindromo
```

El output esperado para un número no palindromo es el siguiente:

```
1
2    Escribe el numero que quieres saber si es palindromo:
3    123456789
4
5
6    El numero 123456789 no es palindromo
```

Problema 2

Escribe un programa que implemente la estructura de datos Stack usando un arreglo de enteros. Se puede asumir que la capacidad máxima del Stack es de 10 enteros.
El stack es una estructura de datos que permite almacenar información. El modo de acceso a los datos es de tipo ultimo en entrar, primero en salir (LIFO por sus siglas en inglés). El manejo de los datos se maneja con dos operaciones llamadas *push* y *pop*, las cuales ingresan y retiran información respectivamente del stack. En la figura 1 se representa visualmente el stack junto a sus operaciones.

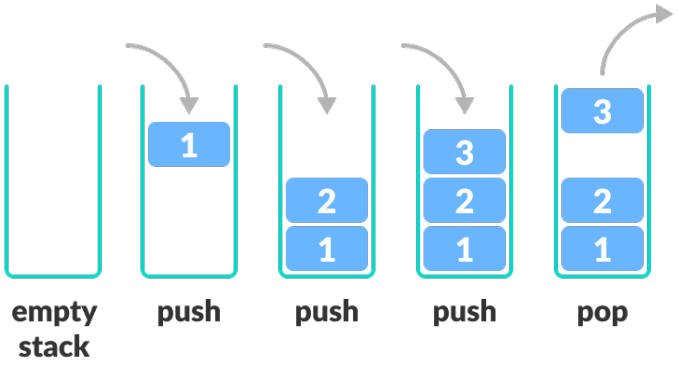

Figura 1: Representación de la estructura de datos stack. Imagen obtenida de Programiz.

El pograma realiza las siguentes acciones:

```
Create stack
Push(2)
Push(3)
Push(4)
Push(5)
Push(6)
Pop
Pop
Pop
Push(4)
Push(5)
```

El output del programa es el siguiente:

```
_____________________________
Push number 2:
2 |
_____________________________
Push number 3:
2 | 3 |
_____________________________
Push number 4:
2 | 3 | 4 |
_____________________________
Push number 5:
2 | 3 | 4 | 5 |
_____________________________
Push number 6:
2 | 3 | 4 | 5 | 6 |
_____________________________
pop:
2 | 3 | 4 | 5 |
_____________________________
pop:
2 | 3 | 4 |
_____________________________
pop:
2 | 3 |
_____________________________
Push number 4:
2 | 3 | 4 |
_____________________________
Push number 5:
2 | 3 | 4 | 5 |
```

El programa se encuentra en la carpeta Problema_2. Para compilarlo se uso el siguiente comando:

```
gcc -Wall -Wextra -Werror -pedantic -ansi -o main.out main.c -std=c11
```

Problema 3

Escribe una función que reciba un arreglo de enteros y que calcule el producto acumulado para cada entrada usando la ecuación 1. Se espera que la complejidad del algoritmo sea lineal O(n). Se puede asumir que la longitud máxima del arreglo será 50.

$$a[i] = \prod_{j \neq i} a[j] \tag{1}$$

Problema 4

Escribe una función que reciba dos arreglos de enteros ordenados en forma no decreciente, y combine dichos arreglos en un solo arreglo también ordenado en forma no decreciente.