Elementos do grupo:

- Diogo Fonseca nº 79858
- Tomás Teodoro nº 80044
- Diogo Silva nº 79828
- Tiago Granja nº 79845

Screenshots do programa a funcionar. O programa é capaz de converter de **qualquer** base para **qualquer** base. É possível notar alguma imprecisão quando as bases tendem a ser muito grandes, sendo impossível anular o erro gerado pela imprecisão gerada quando o computador faz aritmética de *floating point* (embora tenha sido minimizada até certo ponto).

```
PS C:\Users\rokdo\Desktop\Uni_Code\AN_Projects\Decimal_To_Base> python .\any_to_any.py
Select your base: 10
Select the target base: 2
Select the number to convert: -173.8125
-10101101,1101
PS C:\Users\rokdo\Desktop\Uni_Code\AN_Projects\Decimal_To_Base> python .\any_to_any.py
Select your base: 2
Select the target base: 10
Select the number to convert: -10101101.1101
-173,8125
```

```
PS C:\Users\rokdo\Desktop\Uni_Code\AN_Projects\Decimal_To_Base> python .\any_to_any.py
Select your base: 16
Select the target base: 10
Select the number to convert: A3C4.877ED41B
41924,529279
PS C:\Users\rokdo\Desktop\Uni_Code\AN_Projects\Decimal_To_Base> python .\any_to_any.py
Select your base: 10
Select the target base: 16
Select the number to convert: 41924.529279
A3C4,877ED41B
```

```
PS C:\Users\rokdo\Desktop\Uni_Code\AN_Projects\Decimal_To_Base> python .\any_to_any.py
Select your base: 10
Select the target base: 60
Select the number to convert: 4512.21559
1FC,Cu7QNxxP
PS C:\Users\rokdo\Desktop\Uni_Code\AN_Projects\Decimal_To_Base> python .\any_to_any.py
Select your base: 60
Select the target base: 10
Select the number to convert: 1FC.Cu7QNxxP
4512,21559
```

```
Select your base: 10
Select the target base: 100
Select the number to convert: 61398.6182357128359194
6,13,98;61,82,35,71,28,34,57,56
PS C:\Users\rokdo\Desktop\Uni_Code\AN_Projects\Decimal_To_Base> python .\any_to_any.py
Select your base: 100
Select the target base: 10
Select the number to convert: 6,13,98;61,82,35,71,28,34,57,56
61398,61823571
```

```
PS C:\Users\rokdo\Desktop\Uni_Code\AN_Projects\Decimal_To_Base> python .\any_to_any.py
Select your base: 100
Select the target base: 2
Select the number to convert: 13,49;82,81,25
10101000101,110101
```

```
PS C:\Users\rokdo\Desktop\Uni_Code\AN_Projects\Decimal_To_Base> python .\any_to_any.py
Select your base: 2
Select the target base: 16
Select the number to convert: 100101010101110101.0100110101
25575,4D4
PS C:\Users\rokdo\Desktop\Uni_Code\AN_Projects\Decimal_To_Base> python .\any_to_any.py
Select your base: 16
Select the target base: 2
Select the number to convert: 25575.4D4
100101010101110101,01001101
```

```
PS C:\Users\rokdo\Desktop\Uni_Code\AN_Projects\Decimal_To_Base> python .\any_to_any.py
Select your base: 3
Select the target base: 6
Select the number to convert: 120102201.21021002
124031,45014212
PS C:\Users\rokdo\Desktop\Uni_Code\AN_Projects\Decimal_To_Base> python .\any_to_any.py
Select your base: 6
Select the target base: 3
Select the number to convert: 124031.45014212
120102201,21021002
```

Código:

```
import math


def convert_base(x, base):
    y = abs(x)

    if base <= 10:
        y = decimal_to_less_than_10(y, base)
        y = base_round(y, base)
    elif base <= 61:
        y = decimal_to_less_than_62(y, base)
        y = base_round(y, base)
    else:
        y = decimal_to_greater_than_61(y, base)

    if x < 0:
        y = "-" + y

    return y


def decimal_to_less_than_10(x, base):
    whole = decimal_to_less_than_10_whole(x, base)
    fractionary = decimal_to_less_than_10_fractionary(x, base)

    return whole + "," + fractionary


def decimal_to_less_than_10_whole(x, base):
    y = str(int(x))
    if x >= base:
        y = str(decimal_to_less_than_10_whole(x // base, base)) + str(int(x) %
base)
    return y


def decimal_to_less_than_10_fractionary(x, base):
    y = ""
    for i in range(9):
        x -= int(x)
        x = x * base
        y += str(int(x))

    return y


def decimal_to_less_than_62(x, base):
```

```
    whole = decimal_to_less_than_62_whole(x, base)
    fractionary = decimal_to_less_than_62_fractionary(x, base)

    return whole + "," + fractionary


def decimal_to_less_than_62_whole(x, base):
    y = str(int(x))
    if x >= base:
        y = str(decimal_to_less_than_62_whole(x // base, base)) +
int_to_alphanumeric(
            int(x) % base
        )
    else:
        y = int_to_alphanumeric(int(x))
    return y


def decimal_to_less_than_62_fractionary(x, base):
    y = ""
    for i in range(9):
        x -= int(x)
        x = x * base
        y += int_to_alphanumeric(int(x))
    return y


def decimal_to_greater_than_61(x, base):
    whole = decimal_to_greater_than_61_whole(x, base)
    fractionary = decimal_to_greater_than_61_fractionary(x, base)

    return whole + ";" + fractionary


def decimal_to_greater_than_61_whole(x, base):
    y = str(int(x))
    if x >= base:
        y = (
            str(decimal_to_greater_than_61_whole(x // base, base))
            + ","
            + str(int(x) % base)
        )
    return y


def decimal_to_greater_than_61_fractionary(x, base):
    y = ""
    x -= int(x)
    x = x * base
```

```
    y += str(int(x))

    for i in range(7):
        x -= int(x)
        x = x * base
        y += "," + str(int(x))
    return y


def int_to_alphanumeric(x):
    if x >= 10 and x <= 35:
        return chr(x + 55)
    elif x >= 36 and x <= 61:
        return chr(x + 61)
    return str(x)


def alphanumeric_to_int(char):
    ascii_value = ord(char)
    result = 0
    if ascii_value >= 65 and ascii_value <= 90:
        result = int(ascii_value - 55)
    elif ascii_value >= 97 and ascii_value <= 122:
        result = int(ascii_value - 61)
    else:
        result = int(char)

    return result


def base_round(x, base):
    reverse = x[::-1]
    reverse_list = list(reverse)

    reverse_list.pop(0)
    if alphanumeric_to_int(reverse[0]) >= math.ceil(base / 2):
        reverse = reversed_base_string_list_add(reverse_list, base)

    reverse_list = remove_zeros(reverse_list)

    return "".join(reverse_list)[::-1]


def reversed_base_string_list_add(reverse_list, base):
    for i in range(len(reverse_list)):
        if reverse_list[i] == int_to_alphanumeric(base - 1):
            reverse_list[i] = "0"
        else:
            reverse_list[i] = chr(ord(reverse_list[i]) + 1)
```

```
            break
    return reverse_list


def remove_zeros(reverse_list):
    current_char = reverse_list[0]
    while current_char == "0":
        reverse_list.pop(0)
        current_char = reverse_list[0]
    if reverse_list[0] == ",":
        reverse_list.pop(0)
    return reverse_list


# ------------------- to decimal -------------------


def convert_to_decimal(number_str, base):
    number_str_list = [char for char in number_str]
    result = ""
    prepend = ""
    if number_str_list[0] == "-":
        prepend = number_str_list.pop(0)

    if base <= 10:
        result = less_than_10_to_decimal(number_str_list, base)
    elif base <= 61:
        result = less_than_10_to_decimal(number_str_list, base)
    else:
        result = greater_than_61_to_decimal(number_str_list, base)

    return prepend + str(result)


def less_than_10_to_decimal(number_str_list, base):
    whole = less_than_61_to_decimal_whole(number_str_list, base)
    fractionary = less_than_61_to_decimal_fractionary(number_str_list, base)

    return whole + fractionary


def less_than_61_to_decimal_whole(number_str_list, base):
    number_list = []
    for character in number_str_list:
        if character == ".":
            break
        number_list.append(alphanumeric_to_int(character))
    number_list.reverse()
```

```
    k = 0
    result = 0
    for number in number_list:
        result += number * pow(base, k)
        k += 1

    return result


def less_than_61_to_decimal_fractionary(number_str_list, base):
    result = 0.0
    if "." not in number_str_list:
        return result
    number_list = []
    for character in number_str_list[::-1]:
        if character == ".":
            break
        number_list.append(alphanumeric_to_int(character))
    number_list.reverse()

    k = 1
    for number in number_list:
        result += number / pow(base, k)
        k += 1

    return result


def greater_than_61_to_decimal(number_str_list, base):
    whole = greater_than_61_to_decimal_whole(number_str_list, base)
    fractionary = greater_than_61_to_decimal_fractionary(number_str_list,
base)

    return whole + fractionary


def greater_than_61_to_decimal_whole(number_str_list, base):
    number_list = []
    buffer = ""
    for character in number_str_list:
        if character == ",":
            number_list.append(int(buffer))
            buffer = ""
        elif character == ";":
            break
        else:
            buffer += character
    if buffer != "":
        number_list.append(int(buffer))
```

```
    number_list.reverse()

    k = 0
    result = 0
    for number in number_list:
        result += number * pow(base, k)
        k += 1

    return result


def greater_than_61_to_decimal_fractionary(number_str_list, base):
    result = 0.0
    buffer = ""
    if ";" not in number_str_list:
        return result
    number_list = []
    for character in number_str_list[::-1]:
        if character == ",":
            number_list.append(int(buffer[::-1]))
            buffer = ""
        elif character == ";":
            break
        else:
            buffer += character
    if buffer != "":
        number_list.append(int(buffer[::-1]))
    number_list.reverse()

    k = 1
    for number in number_list:
        result += number / pow(base, k)
        k += 1

    return result


base = int(input("Select your base: "))
target_base = int(input("Select the target base: "))
number = input("Select the number to convert: ")
base_10_number = convert_to_decimal(number, base)
print(convert_base(float(base_10_number), target_base))
```