Homework #12

01286121 Computer Programming

Software Engineering Program,

Department of Computer Engineering,

School of Engineering, KMITL

By

68011278 Ananda Stallard

1.

Code:

```
abbreviations = {
    "be": "b",
    "because": "cuz",
    "see": "c",
    "the": "da",
    "okay": "ok",
    "are": "r",
    "you": "u",
    "without": "w/o",
    "why": "y",
    "see you": "cu",
    "ate": "8",
    "great": "gr8",
    "mate": "m8",
    "wait": "w8",
    "later": "l8r",
    "tomorrow": "2mro",
    "for": "4",
    "before": "b4",
    "once": "1ce",
    "and": "&",
    "your": "ur",
    "you're": "ur",
    "as far as I know": "afaik",
    "as soon as possible": "ASAP",
    "at the moment": "atm",
    "be right back": "brb",
    "by the way": "btw",
    "for your information": "FYI",
    "in my humble opinion": "imho",
    "in my opinion": "imo",
    "laughing out loud": "lol",
    "oh my god": "omg",
    "rolling on the floor laughing": "rofl",
    "talk to you later": "ttyl"
}
```

```
def textese(s):
    text = s.lower()

    for phrase, abbr in abbreviations.items():
        if " " in phrase:
            text = text.replace(phrase, abbr)

    words = text.split()
    for i in range(len(words)):
        if words[i] in abbreviations:
            words[i] = abbreviations[words[i]]

    return " ".join(words)

def untextese(s):
    reverse_abbreviation = {v.lower(): k for k, v in abbreviations.items()}

    text = s.lower()

    words = text.split()
    for i in range(len(words)):
        if words[i] in reverse_abbreviation:
            words[i] = reverse_abbreviation[words[i]]

    return " ".join(words)

abbreviated = textese("see you later, I will be right back as soon as possible and also, that's great")
print(f"Abbreviated sentence: {abbreviated}")

un_abbreviated = untextese(abbreviated)
print(f"Un-Abbreviated sentence: {un_abbreviated}")
```

Result:

```
Abbreviated sentence: cu later, i will brb ASAP & also, that's gr8
Un-Abbreviated sentence: see you later, i will be right back as soon as possible and also, that's great
```

2.

Code:

```
def composite(d1, d2):
    d3 = {}

    for k, v in d1.items():
        for i in d2:
            if v == i:
                d3[k] = d2[i]

    return d3

dict1 = {'a': 'p', 'b': 'r', 'c': 'q', 'd': 'p', 'e': 's'}
dict2 = {'p': '1', 'q': '2', 'r': '3'}

new_dict = composite(dict1, dict2)
print(new_dict)
```

Result:

```
{'a': '1', 'b': '3', 'c': '2', 'd': '1'}
```

3.

Code:

```
def product(*args):
    if not args:
        return set()
    if len(args) == 1:
        return {(x, ) for x in args[0]}

    first, *rest = args
    rest_products = product(*rest)

    cartesian = set()

    for i in first:
        for j in rest_products:
            cartesian.add((i, *j))

    return cartesian

s1 = set([1, 2, 3])
```

```
s2 = set(['p', 'q'])
s3 = set(['a', 'b', 'c'])


print(product(s1, s2))


print(product(s1, s2, s3))


print(product(s1))
```

Result:

```
{(2, 'p'), (3, 'p'), (1, 'p'), (2, 'q'), (3, 'q'), (1, 'q')}
{(3, 'q', 'a'), (1, 'p', 'a'), (3, 'q', 'c'), (1, 'p', 'c'), (1, 'q', 'b'), (3, 'p', 'c'), (1, 'q', 'a'), (3, 'p', 'a'), (1, 'q', 'c'),
 (2, 'q', 'b'), (2, 'q', 'c'), (2, 'p', 'b'), (3, 'q', 'b'), (1, 'p', 'b'), (2, 'q', 'a'), (2, 'p', 'a'), (2, 'p', 'c'), (3, 'p', 'b')}
{(1,), (2,), (3,)}
```