Contents

1 Basic Test Results

```
Sun Dec 13 08:17:30 IST 2020
Process Process-225:
Traceback (most recent call last):
  File "/usr/lib/python3.7/multiprocessing/process.py", line 297, in _bootstrap
    self.run()
  File "/usr/lib/python3.7/multiprocessing/process.py", line 99, in run
    self._target(*self._args, **self._kwargs)
  File "/tmp/bodek.KRNToq/intro2cs1/ex7/tsviel/final/testdir/lib/autotest.py", line 74, in wrap
    res=target(*args, **kwargs)
  File "/tmp/bodek.KRNToq/intro2cs1/ex7/tsviel/final/testdir/lib/testrunners.py", line 30, in import_runner
    return print_runner(modulename, fname, args, kwargs, options,tname)
  File "/tmp/bodek.KRNToq/intro2cs1/ex7/tsviel/final/testdir/lib/testrunners.py", line 53, in print_runner
    tname=tname)
  File "/tmp/bodek.KRNToq/intro2cs1/ex7/tsviel/final/testdir/lib/testrunners.py", line 39, in import_runner
    code,res = peel(runners, modulename, fname, args, kwargs)
  File "/tmp/bodek.KRNToq/intro2cs1/ex7/tsviel/final/testdir/lib/testrunners.py", line 7, in peel
    return runners[-1](modulename, fname, args, kwargs,options,runners[:-1])
  File "/tmp/bodek.KRNToq/intro2cs1/ex7/tsviel/final/testdir/lib/testrunners.py", line 12, in base_runner
    return None,func(*args, **kwargs)
  File "/tmp/bodek.KRNToq/intro2cs1/ex7/tsviel/final/testdir/src/ex7.py", line 71, in play_hanoi
    hanoi.move(src, dst)
  File "/tmp/bodek.KRNToq/intro2cs1/ex7/tsviel/final/testdir/lib/ex7tests_full.py", line 134, in move
    raise Exception("Empty src")
Exception: Empty src
Process Process-229:
Traceback (most recent call last):
  File "/usr/lib/python3.7/multiprocessing/process.py", line 297, in _bootstrap
    self.run()
  File "/usr/lib/python3.7/multiprocessing/process.py", line 99, in run
    self._target(*self._args, **self._kwargs)
  File "/tmp/bodek.KRNToq/intro2cs1/ex7/tsviel/final/testdir/lib/autotest.py", line 74, in wrap
    res=target(*args, **kwargs)
  File "/tmp/bodek.KRNToq/intro2cs1/ex7/tsviel/final/testdir/lib/ex7tests_full.py", line 40, in ignorereturn_runner
    resfilter=resfilter,tname=tname)
  File "/tmp/bodek.KRNToq/intro2cs1/ex7/tsviel/final/testdir/lib/testrunners.py", line 39, in import_runner
    code,res = peel(runners, modulename, fname, args, kwargs)
  File "/tmp/bodek.KRNToq/intro2cs1/ex7/tsviel/final/testdir/lib/testrunners.py", line 7, in peel
    return runners[-1](modulename, fname, args, kwargs,options,runners[:-1])
  File "/tmp/bodek.KRNToq/intro2cs1/ex7/tsviel/final/testdir/lib/testrunners.py", line 12, in base_runner
    return None,func(*args, **kwargs)
  File "/tmp/bodek.KRNToq/intro2cs1/ex7/tsviel/final/testdir/src/ex7.py", line 71, in play_hanoi
    hanoi.move(src, dst)
  File "/tmp/bodek.KRNToq/intro2cs1/ex7/tsviel/final/testdir/lib/ex7tests_full.py", line 134, in move
    raise Exception("Empty src")
Exception: Empty src
Sun Dec 13 08:17:30 IST 2020
Archive:  /tmp/bodek.KRNToq/intro2cs1/ex7/tsviel/final/submission
  inflating: src/ex7.py
12 passed tests out of 12 in test set named 'printton'.
result_code    printton    12    1
200 passed tests out of 200 in test set named 'digitsum'.
result_code    digitsum    200    1
12 passed tests out of 12 in test set named 'isprime'.
result_code    isprime    12    1
--> BEGIN TEST INFORMATION
Test name: hanoi_0
Module tested: ex7
Function call: play_hanoi([[], [], []],0,0,1,2)
Expected return value: None
```

```
Expected print string: ''
--> END TEST INFORMATION
************************************************************************
********************     There is a problem:
********************     The test named 'hanoi_0' failed.
************************************************************************
Test did not complete, exited with exitcode 1.
This probably means your code caused an exception to be raised.
result_code    hanoi_0    exception    1
--> BEGIN TEST INFORMATION
Test name: hanoi_0b
Module tested: ex7
Function call: play_hanoi([[], [], []],0,0,1,2)
Expected return value: ''
--> END TEST INFORMATION
************************************************************************
********************     There is a problem:
********************     The test named 'hanoi_0b' failed.
************************************************************************
Test did not complete, exited with exitcode 1.
This probably means your code caused an exception to be raised.
result_code    hanoi_0b    exception    1
6 passed tests out of 8 in test set named 'hanoi'.
result_code    hanoi    6    1
12 passed tests out of 12 in test set named 'printseq'.
result_code    printseq    12    1
10 passed tests out of 10 in test set named 'printnorep'.
result_code    printnorep    10    1
8 passed tests out of 8 in test set named 'parentheses'.
result_code    parentheses    8    1
10 passed tests out of 10 in test set named 'floodfill'.
result_code    floodfill    10    1
3 passed tests out of 3 in test set named 'recursion'.
result_code    recursion    3    1
Running mypy
src/ex7.py:125: error: Missing type parameters for generic type
src/ex7.py:130: warning: Returning Any from function declared to return "List[str]"
src/ex7.py:142: error: Need type annotation for 'chest'
src/ex7.py:146: error: Missing type parameters for generic type
src/ex7.py:146: error: Missing return statement
src/ex7.py:150: error: Return value expected
Warning: mypy should produce no output and complete successfully!
Finished running mypy

TESTING COMPLETED
```

2 ex7.py

```python
##################################################################
# FILE : ex7.py
# WRITER : TSVIEL ZAIKMAN , tsviel , 208241133
# EXERCISE : intro2cs ex7 2020
# DESCRIPTION: Recursion Exercise
##################################################################

from typing import List, Dict, Tuple, Union, Any

# Custom Typings
CHAR_LIST = List[str]
CHAR_MATRIX = List[CHAR_LIST]

# Operators for the recursive function
ALLOW_REPEATS = True  # To allow repeats
RESTRICT_REPEATS = False  # To disallow repeats

# Fillers for Flood fill Function
FILL = "*"


##################################################################
# Part 1
##################################################################

def print_to_n(n: int) -> None:
    """The function prints all Natural numbers from 1 to n"""
    if n < 1:
        return None
    print_to_n(n - 1)
    print(n)


def digit_sum(n: int) -> int:
    """This function sums the digits of an integer"""
    if n == 0:
        return 0
    return n % 10 + digit_sum(int(n / 10))


def has_divisor_smaller_than(n: int, i: int) -> bool:
    """If n(Integer), has divisor smaller than i(Integer) returns True,
    Otherwise returns False"""
    # Base Cases
    if n < 2:
        return False
    if n == 2:
        return True
    if n % i == 0:
        return False

    if (i * i) > n:
        return True
    return has_divisor_smaller_than(n, i + 1)


def is_prime(n: int) -> bool:
    """Returns True if n(integer) is a prime number, False if not"""
    if has_divisor_smaller_than(n, 2):
```

```
60              return True
61          return False
62
63
64      ###################################################################
65      # Part 2
66      ###################################################################
67
68      def play_hanoi(hanoi: Any, n: int, src: Any, dst: Any, temp: Any) -> None:
69          """The function solves Hanoi Game"""
70          if n < 1:  # Extreme Case
71              hanoi.move(src, dst)
72              return
73          if n == 1:  # Base Case
74              hanoi.move(src, dst)
75          else:  # Recursion Step
76              play_hanoi(hanoi, n - 1, src, temp, dst)
77              play_hanoi(hanoi, 1, src, dst, temp)
78              play_hanoi(hanoi, n - 1, temp, dst, src)
79
80
81      def print_sequences(char_list: CHAR_LIST, n: int) -> None:
82          """The Function prints all possible n length combinations from
83          a list of chars recursively with repeats"""
84          length = len(char_list)
85          print_sequences_recursively(char_list, "", length, n, ALLOW_REPEATS)
86
87
88      def print_no_repetition_sequences(char_list: CHAR_LIST, n: int) -> None:
89          """The Function prints all possible n length combinations from
90          a list of chars recursively without repeats"""
91          length = len(char_list)
92          print_sequences_recursively(char_list, "", length, n, RESTRICT_REPEATS)
93
94
95      def print_sequences_recursively(char_list: CHAR_LIST, prefix: str, n: int,
96                                      length: int, reps: int) -> None:
97          """The function handles the recursive combination creation of letters"""
98          if reps:  # Check if we are allowed to repeat letters, default is True
99              if length == 0:  # print prefix
100                 print(prefix)
101                 return
102             # One by one add all characters from set
103             # recursively call for the length equals to length-1
104             for i in range(n):  # Next character of input added
105                 new_prefix = prefix + char_list[i]
106                 print_sequences_recursively(char_list, new_prefix, n, length - 1,
107                                             ALLOW_REPEATS)
108         elif not reps:
109             if length == 0:  # print prefix
110                 print(prefix)
111                 return
112             # One by one add all characters from set and recursively call for
113             # length equals to length-1
114             for i in range(n):  # Next character of input added
115                 new_prefix = prefix + char_list[i]
116                 # the string length is decreased, because we have added a new char
117                 # Checks if any of the letters inside the char_list appears for
118                 # each Iteration we make on char_list
119                 if any(char not in prefix for char in char_list[i]):
120                     print_sequences_recursively(char_list,
121                                                 new_prefix, n, length - 1,
122                                                 RESTRICT_REPEATS)
123
124
125     def _parentheses(n: int, chest: Dict) -> CHAR_LIST:
126         """The recursive function of Parentheses"""
127         if n == 0:  # Base case of 0
```

```
128             return sorted([''])  # Return a list of the function
129         elif n in chest:
130             return chest[n]  # Update the memory chest
131         else:
132             output = set('(' + p + ')' for p in parentheses(n - 1))
133             for k in range(1, n):  # Iteration for the index range from 1 to n
134                 output.update(
135                     p + q for p in parentheses(k) for q in parentheses(n - k))
136             chest[n] = output  # Append the cached memory to output
137             return sorted(output)  # Return a list of the function
138 
139 
140 def parentheses(n: int) -> CHAR_LIST:
141     """The function change the parenthesis according to given coordinates"""
142     chest = {}  # A chest(Caching) for the parentheses function
143     return _parentheses(n, chest)
144 
145 
146 def flood_fill(image: CHAR_MATRIX, start: Tuple) -> Union[CHAR_MATRIX, None]:
147     """Flood Fill Function to replace . with * according to coordinates"""
148     x, y = start  # Assign the values of the tuple to x,y coordinates
149     if image[x][y] == FILL:  # If the place already filled
150         return
151     image[x][y] = FILL  # Fill the current cell
152     flood_fill(image, (x + 1, y))  # Draw up
153     flood_fill(image, (x - 1, y))  # Draw Down
154     flood_fill(image, (x, y + 1))  # Draw Right
155     flood_fill(image, (x, y - 1))  # Draw left
```