```
from io import StringIO
import unittest
from unittest.mock import patch, mock_open
import main1

class TestProgram(unittest.TestCase):

    @patch("builtins.open", mock_open(read_data="test data"))
    def test_filter_details_ending_with_st(self):
        expected_result = [("Schindler's List", 'Central Library'), ("One
Flew Over the Cuckoo's Nest", 'West Library')]
        with patch("sys.stdout", new_callable=StringIO) as mock_stdout:
            main1.main()
            actual_output = mock_stdout.getvalue()
            self.assertTrue(all(str(detail) in actual_output for detail in
expected_result))

    @patch("builtins.open", mock_open(read_data="test data"))
    def test_average_prices_by_lib(self):
        expected_result = [('East Library', 768), ('West Library', 1536),
('North Library', 1536), ('South Library', 4096), ('Central Library',
8362)]
        with patch("sys.stdout", new_callable=StringIO) as mock_stdout:
            main1.main()
            actual_output = mock_stdout.getvalue()
            self.assertTrue(all(str(provider) in actual_output for provider
in expected_result))

    @patch("builtins.open", mock_open(read_data="test data"))
    def test_filter_libs_starting_with_c(self):
        expected_result = {'Central Library': ['The Godfather',
"Schindler's List", 'The Godfather Part II']}
        with patch("sys.stdout", new_callable=StringIO) as mock_stdout:
            main1.main()
            actual_output = mock_stdout.getvalue()
            self.assertTrue(all(LibOfCD in actual_output for LibOfCD in
expected_result.keys()))
            self.assertTrue(all(all(CD_Disk in actual_output for CD_Disk in
CD_Disks) for LibOfCD, CD_Disks in expected_result.items()))

if __name__ == '__main__':
    unittest.main()
```