Міністерство освіти і науки України

Національний університет «Запорізька Політехніка»

Кафедра програмних засобів

ЗВІТ

з лабораторної роботи №2

з дисципліни «Спортивне програмування» на тему:

«Алгоритми довгої арифметики»

Виконав:

Студент групи КНТ-122 О. А. Онищенко

Прийняли:

Викладач: С. Д. Леощенко

2023

Рекурентні послідовності

Мета роботи

Вивчити основні можливості та принципи роботи із довгою арифметикою.

Завдання до роботи

Ознайомитися з основними теоретичними відомостями за темою роботи, використовуючи ці методичні вказівки, а також рекомендовану літературу.

Обрати та виконати дві задачі із запропонованого переліку.

Скласти програму обчислення точного значення n!, де n>12.

Скласти програму обчислення точного значення n^n, де n>10.

Обчислити 100!+2^100.

Обчислити 100!−2^100.

Обчислити 7^123

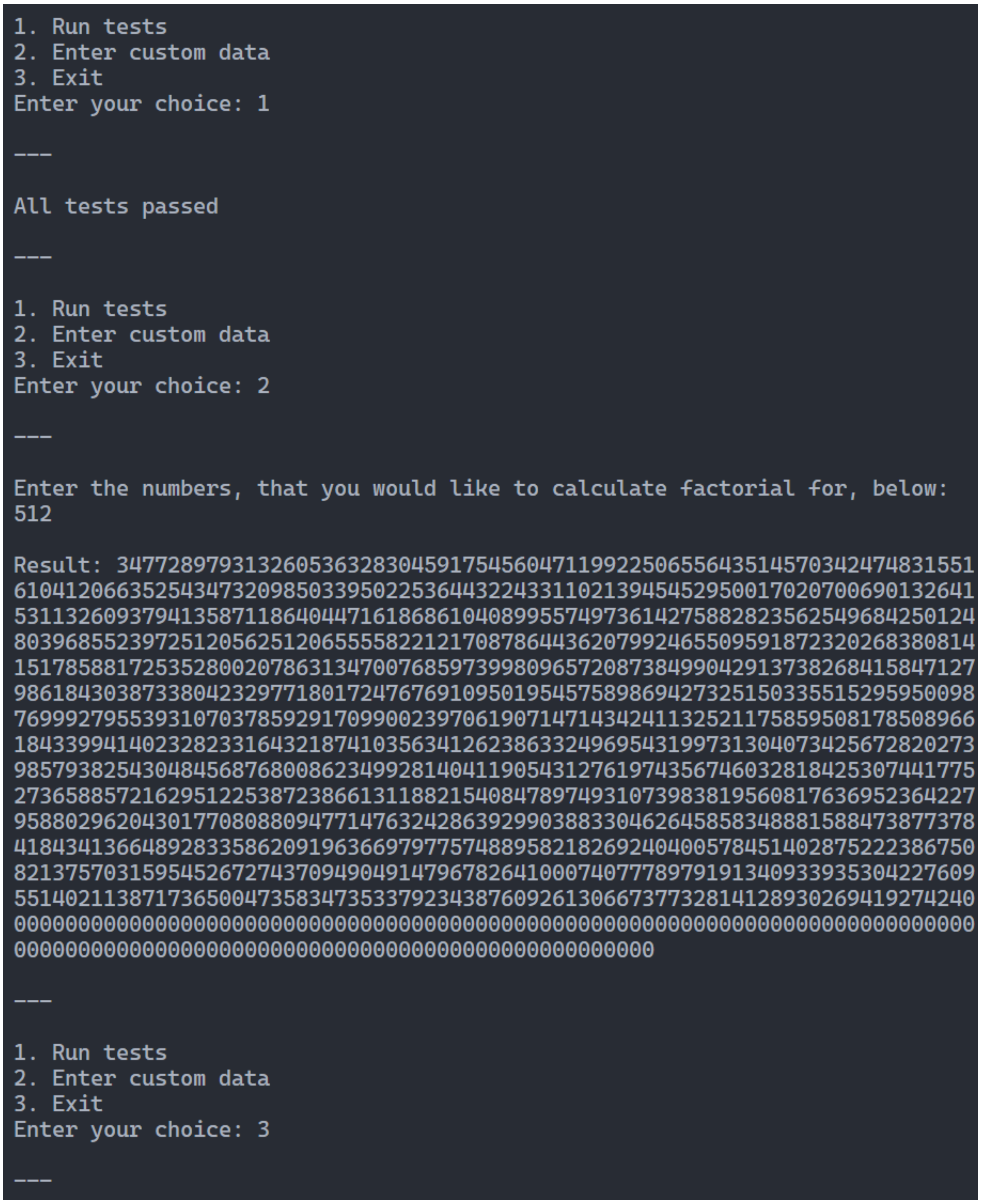
З'ясувати, яке з чисел a^m чи b^n – більше і на скільки (за умови, що a,b≤40000, а m,n≤10)

Скласти програму обчислення точного значення суми 1!+2!+3!+⋯+n! при умові, що n>10

Обчислити точне значення суми 1^2+2^2+3^2+⋯+n^2при умові, що n≥400

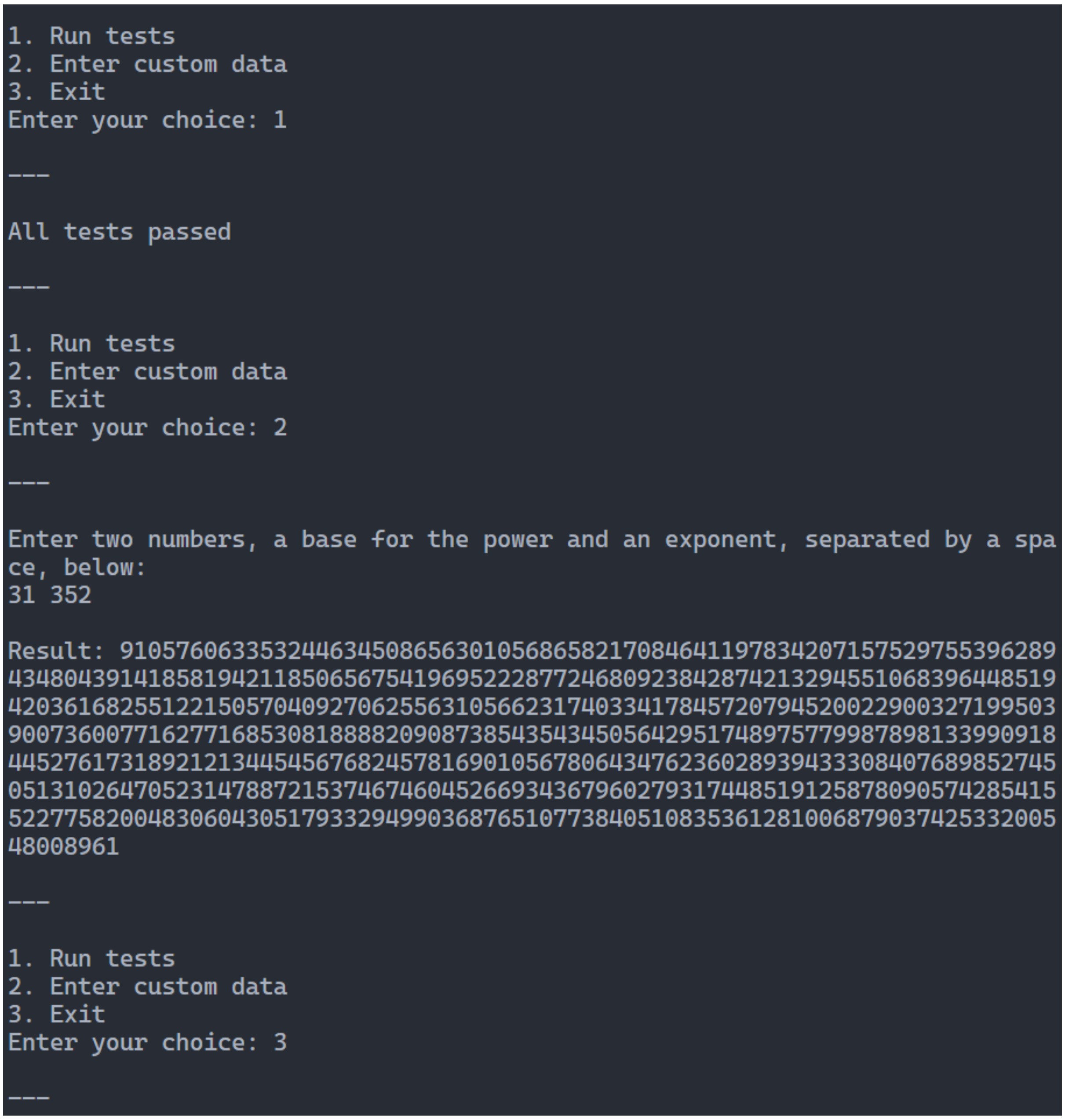
Обчислити точне значення суми 1^n+2^n+3^n+⋯+n^n при умові, що n≥10

Результати виконання



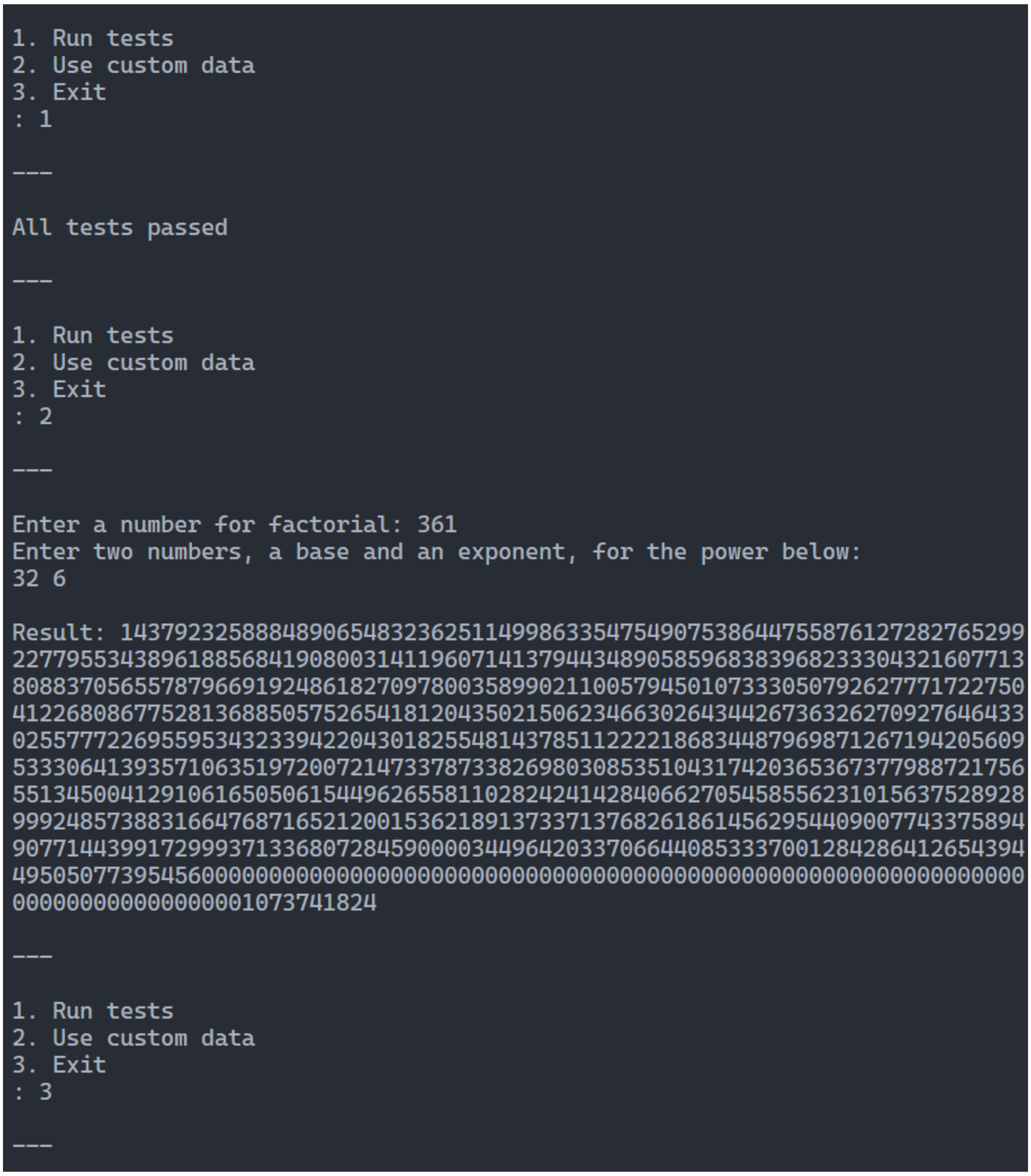
```
1. Run tests
2. Enter custom data
3. Exit
Enter your choice: 1

---

All tests passed

---

1. Run tests
2. Enter custom data
3. Exit
Enter your choice: 2

---

Enter the numbers, that you would like to calculate factorial for, below:
512

Result: 3477289793132605363283045917545604711992250655643514570342474831551
610412066352543473209850339502253644322433110213945452950017020700690132641
531132609379413587118640447161868610408995574973614275882823562549684250124
803968552397251205625120655558221217087864436207992465509591872320268380814
151785881725352800207863134700768597399809657208738499042913738268415847127
986184303873380423297718017247676910950195457589869427325150335515295950098
769992795539310703785929170990023970619071471434241132521175859508178508966
184339941402328233164321874103563412623863324969543199731304073425672820273
985793825430484568768008623499281404119054312761974356746032818425307441775
273658857216295122538723866131188215408478974931073983819560817636952364227
958802962043017708088094771476324286392990388330462645858348881588473877378
418434136648928335862091963669797757488958218269240400578451402875222386750
821375703159545267274370949049147967826410007407778979191340933935304227609
551402113871736500473583473533792343876092613066737732814128930269419274240
000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000

---

1. Run tests
2. Enter custom data
3. Exit
Enter your choice: 3

---
```

Результати виконання програми 1



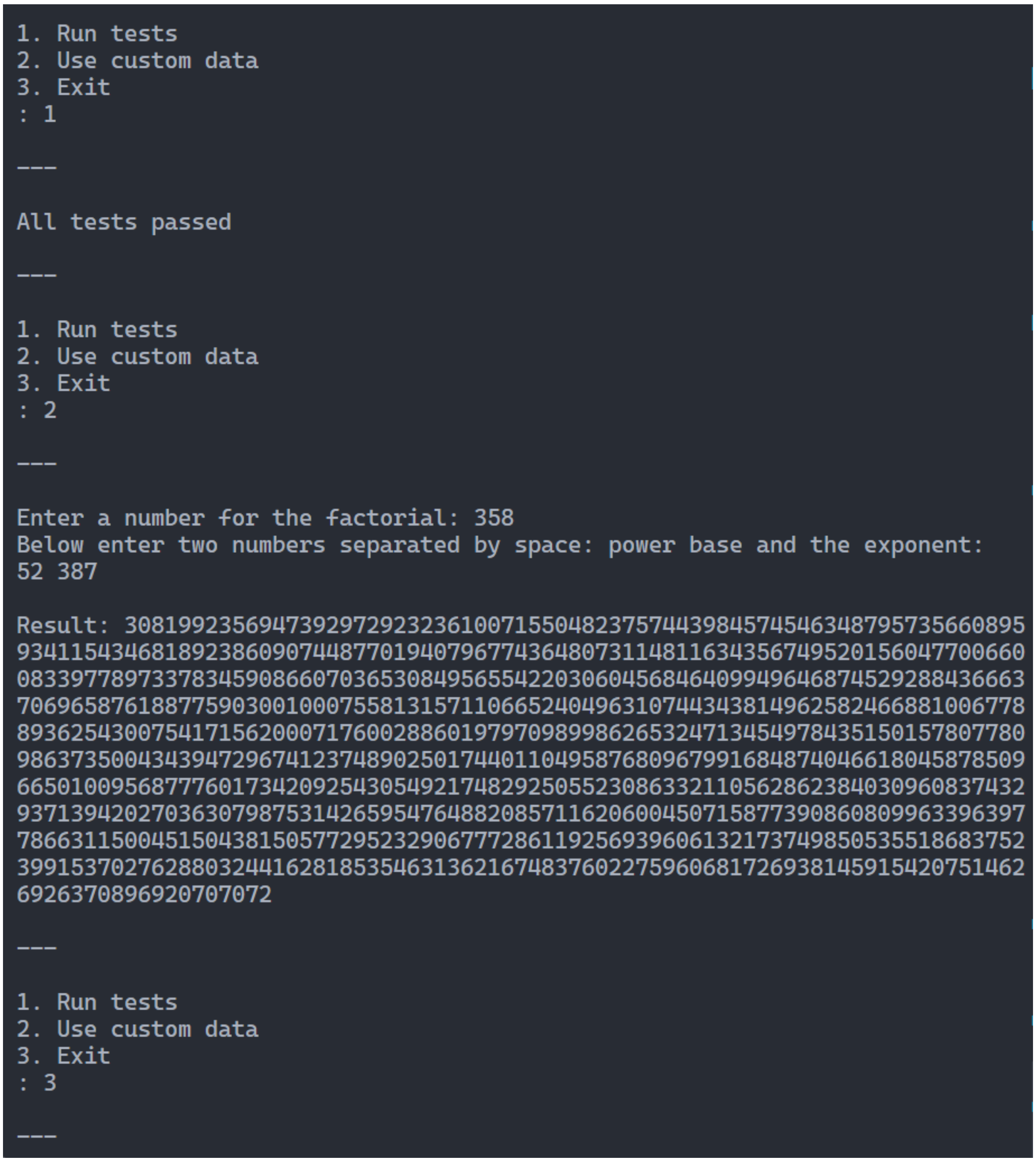
```
1. Run tests
2. Enter custom data
3. Exit
Enter your choice: 1

---

All tests passed

---

1. Run tests
2. Enter custom data
3. Exit
Enter your choice: 2

---

Enter two numbers, a base for the power and an exponent, separated by a spa
ce, below:
31 352

Result: 91057606335324463450865630105686582170846411978342071575297553962889
4348043914185819421185065675419695222877246809238428742132945510683964448519
4203616825512215057040927062556310566231740334178457207945200229003271995503
9007360077162771685308188882090873854354345056429517489757799878981339909918
4452761731892121344545676824578169010567806434762360289394333084076898527445
0513102647052314788721537467460452669343679602793174485191258780905742854415
5227758200483060430517933294990368765107738405108353612810068790374253320005
48008961

---

1. Run tests
2. Enter custom data
3. Exit
Enter your choice: 3

---
```

Результати виконання програми 2



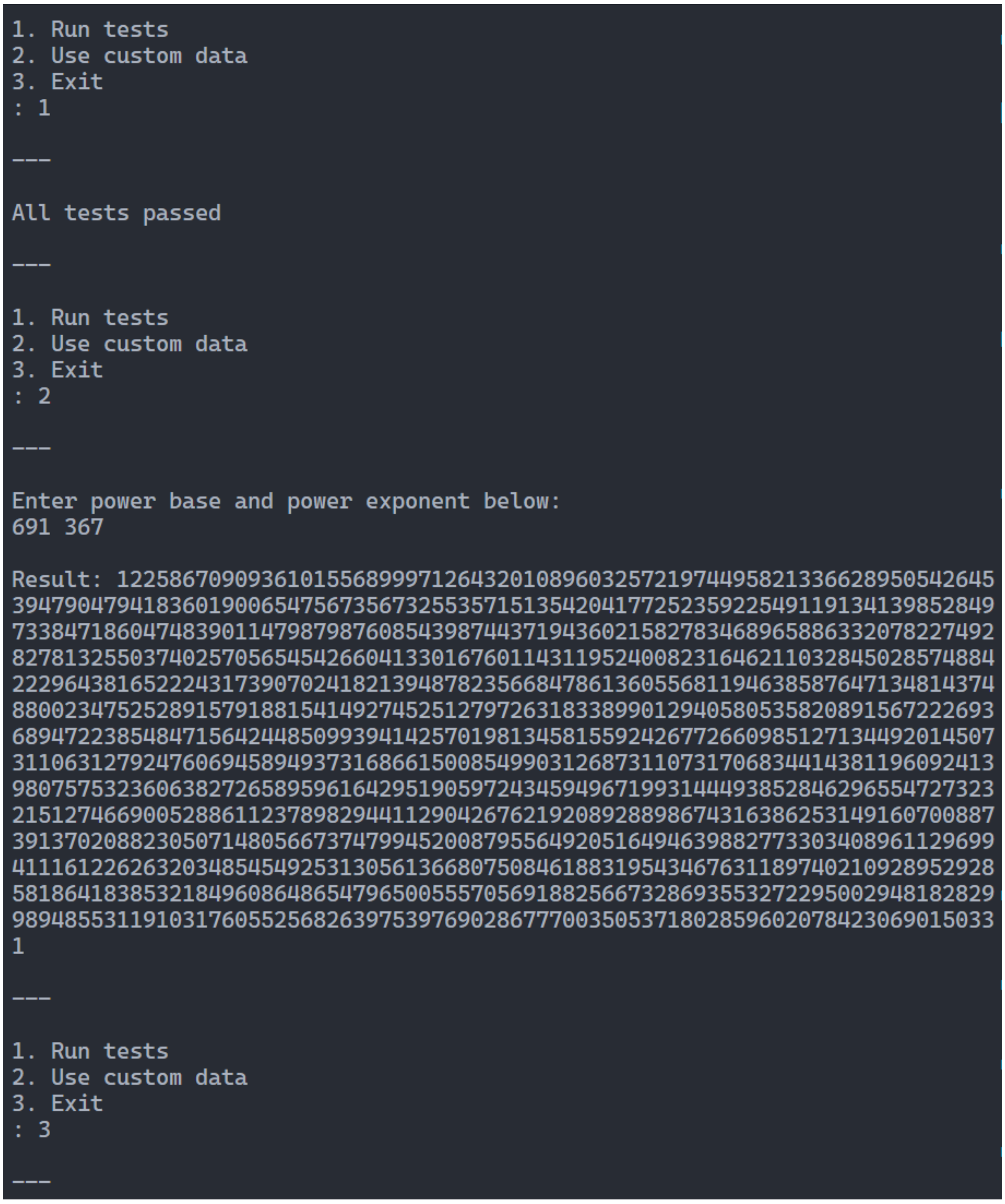
```
1. Run tests
2. Use custom data
3. Exit
: 1

---

All tests passed

---

1. Run tests
2. Use custom data
3. Exit
: 2

---

Enter a number for factorial: 361
Enter two numbers, a base and an exponent, for the power below:
32 6

Result: 1437923258884890654832362511499863354754907538644755876127282765299
227795534389618856841908003141196071413794434890585968383968233304321607713
808837056557879669192486182709780035899021100579450107333050792627771722750
412268086775281368850575265418120435021506234663026434426736326270927646433
025577722695595343233942204301825548143785112222186834487969871267194205609
533306413935710635197200721473378733826980308535104317420365367377988721756
551345004129106165050615449626558110282424142840662705458556231015637528928
999248573883166476871652120015362189137337137682618614562954409007743375894
907714439917299937133680728459000034496420337066440853337001284286412654394
495050773954560000000000000000000000000000000000000000000000000000000000000
0000000000000000001073741824

---

1. Run tests
2. Use custom data
3. Exit
: 3

---
```

Результати виконання програми 3



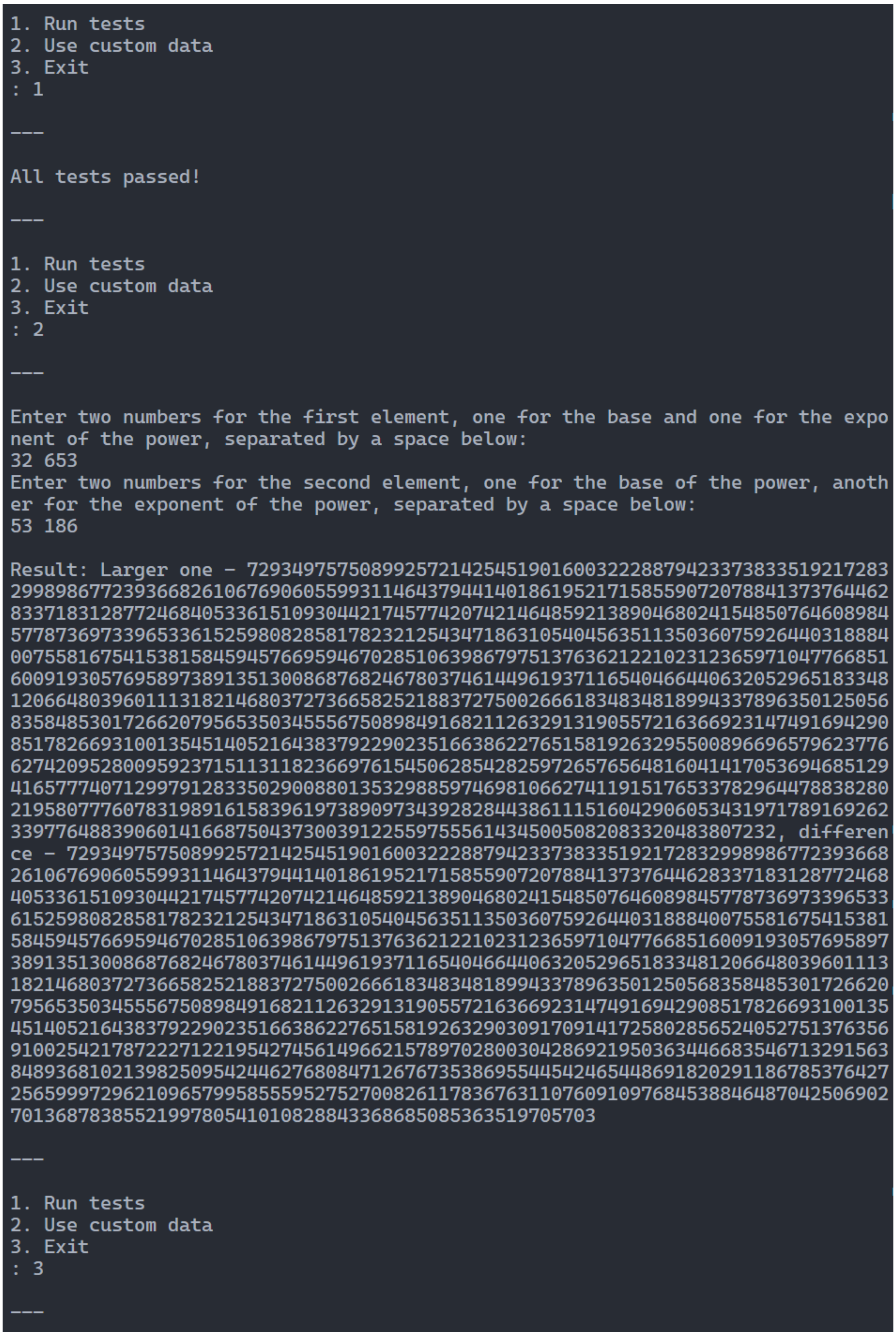
```
1. Run tests
2. Use custom data
3. Exit
: 1

---

All tests passed

---

1. Run tests
2. Use custom data
3. Exit
: 2

---

Enter a number for the factorial: 358
Below enter two numbers separated by space: power base and the exponent:
52 387

Result: 30819923569473929729232361007155048237574439845745463487957356608950
93411543468189238609074487701940796774364807311481163435674952015604770066
08339778973378345908660703653084956554220306045684640994964687452928843666
37069658761887759030010007558131571106652404963107443438149625824668810067
78893625430075417156200071760028860197970989986265324713454978435150157807
780986373500434394729674123748902501744011049587680967991684874046618045878
509665010095687776017342092543054921748292505523086332110562862384030960837
432937139420270363079875314265954764882085711620600450715877390860809963396
397786631150045150438150577295232906777286119256939606132173749850535518683
752399153702762880324416281853546313621674837602275960681726938145915420751
4626926370896920707072

---

1. Run tests
2. Use custom data
3. Exit
: 3

---
```

Результати виконання програми 4



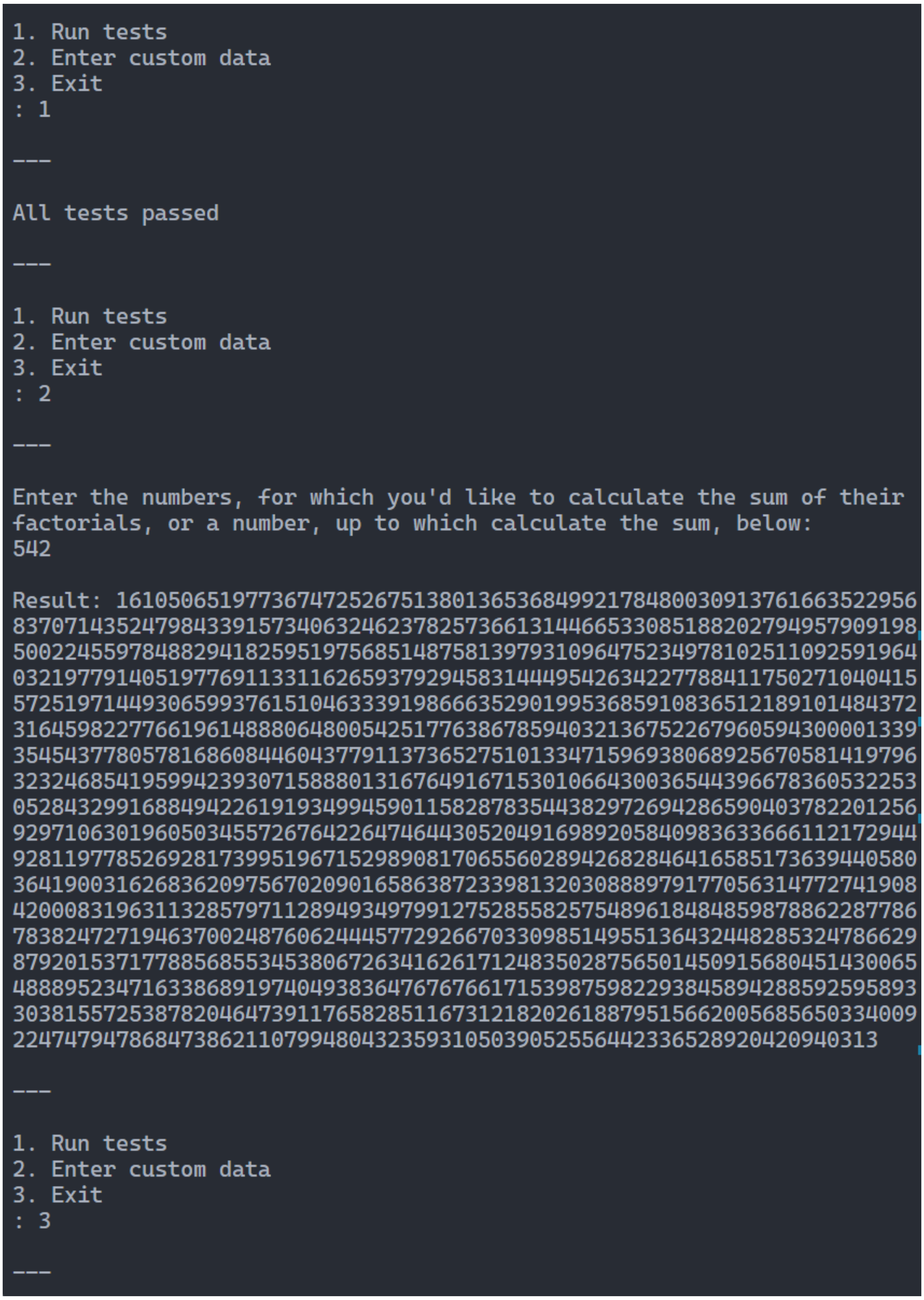
```
1. Run tests
2. Use custom data
3. Exit
: 1

---

All tests passed

---

1. Run tests
2. Use custom data
3. Exit
: 2

---

Enter power base and power exponent below:
691 367

Result: 12258670909361015568999712643201089603257219744958213366289505426 45
394790479418360190065475673567325535715135420417725235922549119134139852849
733847186047483901147987987608543987443719436021582783468965886332078227492
827813255037402570565454266041330167601143119524008231646211032845028574884
222964381652224317390702418213948782356684786136055681194638587647134814374
880023475252891579188154149274525127972631833899012940580535820891567222693
689472238548471564244850993941425701981345815592426772660985127134492014507
311063127924760694589493731686615008549903126873110731706834414381196092413
980757532360638272658959616429519059724345949671993144493852846296554727323
215127466900528861123789829441129042676219208928898674316386253149160700887
391370208823050714805667374799452008795564920516494639882773303408961129699
411161226263203485454925313056136680750846188319543467631189740210928952928
581864183853218496086486547965005557056918825667328693553272295002948182829
989485531191031760552568263975397690286777003505371802859602078423069015033
1

---

1. Run tests
2. Use custom data
3. Exit
: 3

---
```

Результати виконання програми 5



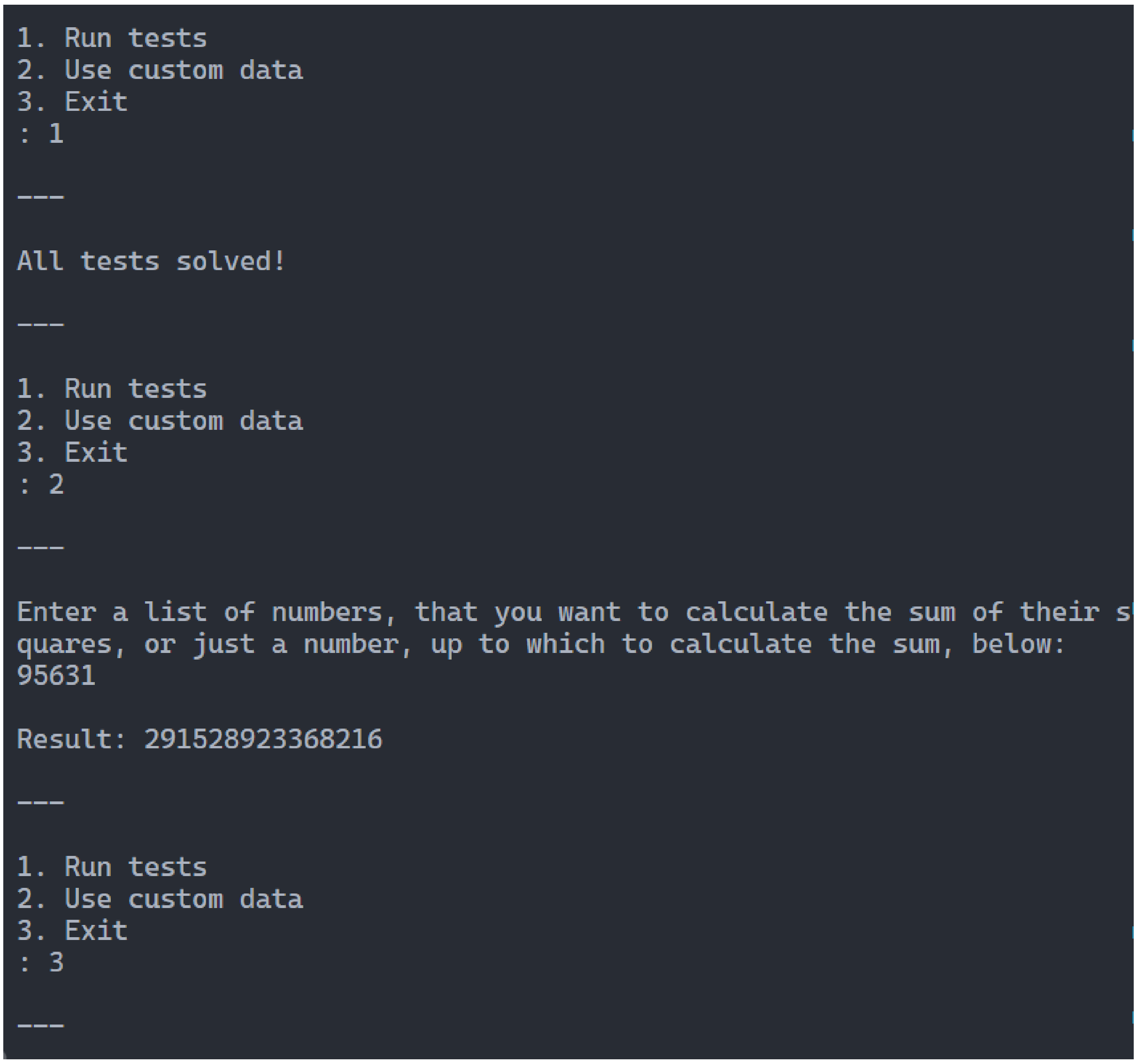
```
1. Run tests
2. Use custom data
3. Exit
: 1

---

All tests passed!

---

1. Run tests
2. Use custom data
3. Exit
: 2

---

Enter two numbers for the first element, one for the base and one for the expo
nent of the power, separated by a space below:
32 653
Enter two numbers for the second element, one for the base of the power, anoth
er for the exponent of the power, separated by a space below:
53 186

Result: Larger one - 729349757508992572142545190160032228879423373833519217283
```

```
Result: Larger one - 729349757508992572142545190160032228879423373833519217283
299898677239366826106769060559931146437944140186195217158559072078841373764462
833718312877246840533615109304421745774207421464859213890468024154850764608984
577873697339653361525980828581782321254347186310540456351135036075926440318884
007558167541538158459457669594670285106398679751376362122102312365971047766851
600919305769589738913513008687682467803746144961937116540466440632052965183348
120664803960111318214680372736658252188372750026661834834818994337896350125056
835848530172662079565350345556750898491682112632913190557216366923147491694290
851782669310013545140521643837922902351663862276515819263295500896696579623776
627420952800959237151131182366976154506285428259726576564816041417053694685129
416577740712997912833502900880135329885974698106627411915176533782964478838280
219580777607831989161583961973890973439282844386111516042906053431971789169262
33977648839060141668750437300391225597555614345005082083320483807232, differen
```



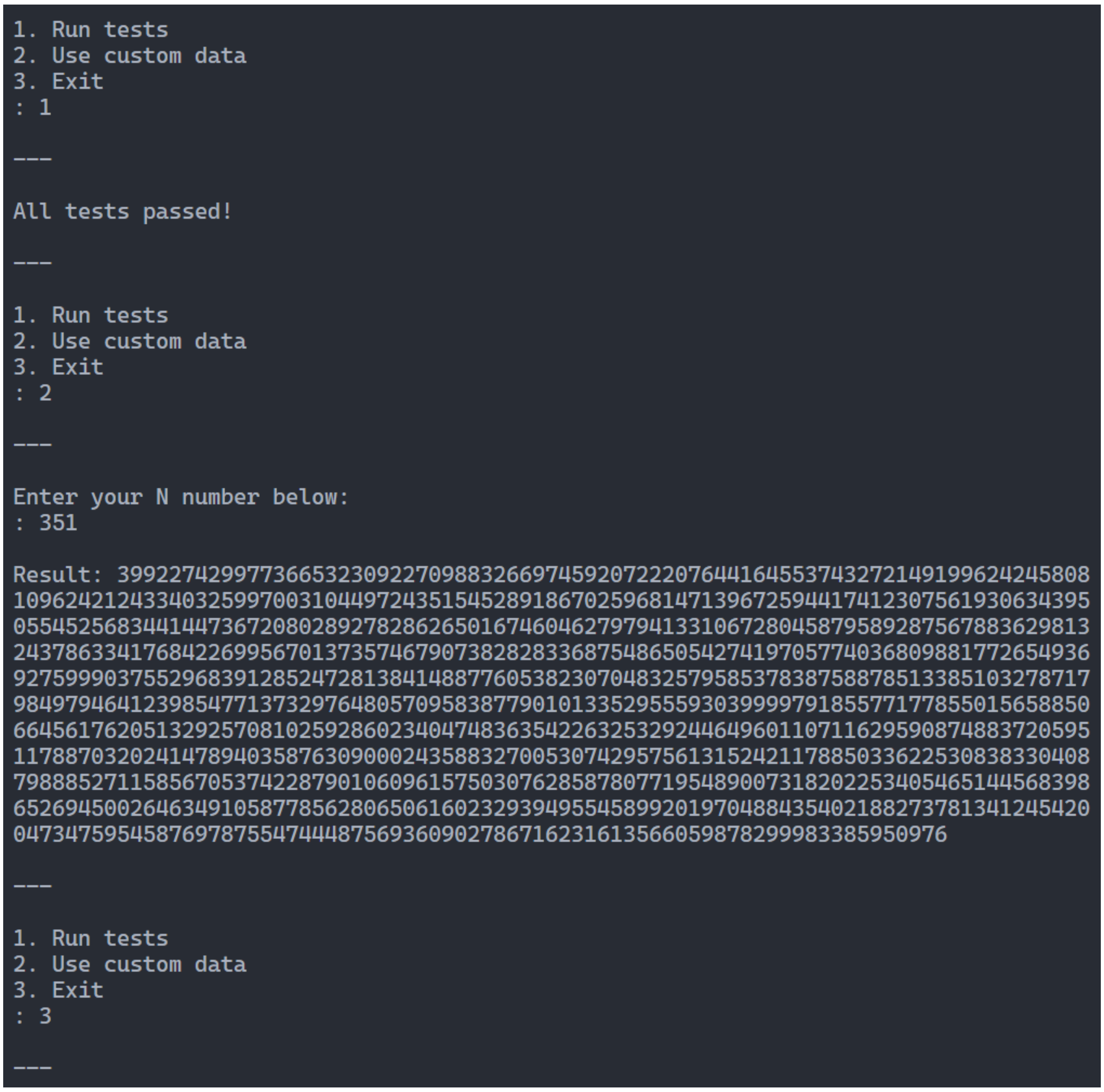
```
1. Run tests
2. Enter custom data
3. Exit
: 1

---

All tests passed

---

1. Run tests
2. Enter custom data
3. Exit
: 2

---

Enter the numbers, for which you'd like to calculate the sum of their
factorials, or a number, up to which calculate the sum, below:
542

Result: 16105065197736747252675138013653684992178480030913761663522956
8370714352479843391573406324623782573661314466533085188202794957909198
5002245597848829418259519756851487581397931096475234978102511092591964
0321977914051977691133116265937929458314449542634227788411750271040415
5725197144930659937615104633391986663529019953685910836512189101484372
3164598227766196148880648005425177638678594032136752267960594300001339
3545437780578168608446043779113736527510133471596938068925670581419796
3232468541959942393071588801316764916715301066430036544396678360532253
0528432991688494226191934994590115828783544382972694286590403782201256
9297106301960503455726764226474644305204916989205840983633666112172944
9281197785269281739951967152989081706556028942682846416585173639440580
3641900316268362097567020901658638723398132030888979177056314772741908
4200083196311328579711289493497991275285582575489618484859878862287786
7838247271946370024876062444577292667033098514955136432448285324786629
8792015371778856855345380672634162617124835028756501450915680451430065
4888952347163386891974049383647676766171539875982293845894288592595893
3038155725387820464739117658285116731218202618879515662005685650334009
2247479478684738621107994804323593105039052556442336528920420940313

---

1. Run tests
2. Enter custom data
3. Exit
: 3

---
```

Результати виконання програми 7



```
1. Run tests
2. Use custom data
3. Exit
: 1

---

All tests solved!

---

1. Run tests
2. Use custom data
3. Exit
: 2

---

Enter a list of numbers, that you want to calculate the sum of their s
quares, or just a number, up to which to calculate the sum, below:
95631

Result: 291528923368216

---

1. Run tests
2. Use custom data
3. Exit
: 3

---
```

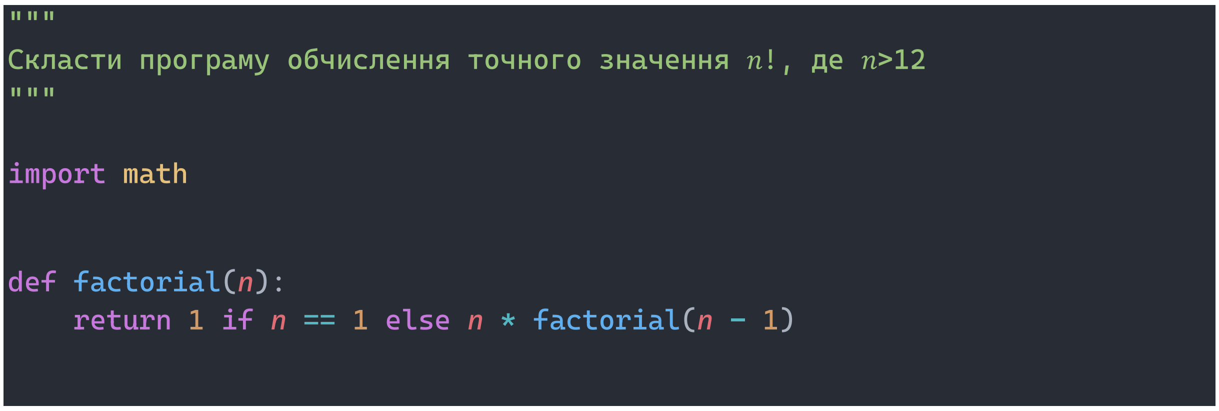
Результати виконання програми 8



```
1. Run tests
2. Use custom data
3. Exit
: 1

---

All tests passed!

---

1. Run tests
2. Use custom data
3. Exit
: 2

---

Enter your N number below:
: 351

Result: 399227429977366532309227098832669745920722207644164553743272149199624245808
```

Результати виконання програми 9

Програмний код



```python
"""
Скласти програму обчислення точного значення n!, де n>12
"""

import math


def factorial(n):
    return 1 if n == 1 else n * factorial(n - 1)
```

```python
def tests():
    assert factorial(13) == math.factorial(13), "Test 1 failed"
    assert factorial(26) == math.factorial(26), "Test 2 failed"
    assert factorial(61) == math.factorial(61), "Test 3 failed"
    assert factorial(256) == math.factorial(256), "Test 4 failed"
    assert factorial(128) == math.factorial(128), "Test 5 failed"
    assert factorial(912) == math.factorial(912), "Test 6 failed"
    print("All tests passed")


def main():
    print()
    while True:
        print("1. Run tests")
        print("2. Enter custom data")
        print("3. Exit")
        choice = input("Enter your choice: ")
        print("\n---\n")
        if choice == "1":
            tests()
        elif choice == "2":
            print(
                f"Enter the numbers, that you would like to calculate
factorial for, below:"
            )
            num = int(input())
            res = factorial(num)
            print(f"\nResult: {res}")
        else:
            break
        print("\n---\n")


if __name__ == "__main__":
    main()
```

```python
"""
Скласти програму для обчислення точного значення n**k, де n>10
"""


def pow(n, power):
    res = 1
    for _ in range(power):
        res *= n
    return res
```

```python
def tests():
    assert pow(13, 13) == 13**13, "Test 1 failed"
    assert pow(51, 51) == 51**51, "Test 2 failed"
    assert pow(318, 318) == 318**318, "Test 3 failed"
    assert pow(916, 916) == 916**916, "Test 4 failed"
    print("All tests passed")


def main():
    print()
    while True:
        print("1. Run tests")
        print("2. Enter custom data")
        print("3. Exit")
        choice = input("Enter your choice: ")
        print("\n---\n")
        if choice == "1":
            tests()
        elif choice == "2":
            print(f"Enter two numbers, a base for the power and an
exponent, separated by a space, below:")
            num, power = list(map(int, input().split()))
            res = pow(num, power)
            print(f"\nResult: {res}")
        else:
            break
        print("\n---\n")


if __name__ == "__main__":
    main()
```

```python
"""
Обчислити 100! + 2^100
"""

import math


def solve(factorialNumber=100, powerBase=2, powerExponent=100):
    return factorial(factorialNumber) + pow(powerBase, powerExponent)


def factorial(n):
    return 1 if n == 1 else n * factorial(n - 1)


def pow(n, power):
    return 1 if power == 0 else n * pow(n, power - 1)


def tests():
    assert solve() == math.factorial(100) + 2**100
    print("All tests passed")


def main():
    print()
    while True:
        print("1. Run tests")
        print("2. Use custom data")
        print("3. Exit")
        choice = int(input(": "))
        print(f"\n---\n")
        if choice == 1:
            tests()
        elif choice == 2:
            factorialNumber = int(input(f"Enter a number for factorial: "))
            print(f"Enter two numbers, a base and an exponent, for the power below:")
            powerBase, powerExponent = list(map(int, input().split()))
            res = solve(factorialNumber, powerBase, powerExponent)
            print(f"\nResult: {res}")
        else:
            break
        print("\n---\n")


if __name__ == "__main__":
    main()
```

```python
"""
Обчислити 100! - 2^100
"""

import math


def solve(factorialNumber=100, powerBase=2, powerExponent=100):
    return factorial(factorialNumber) - pow(powerBase, powerExponent)


def factorial(n):
    return 1 if n == 1 else n * factorial(n - 1)
```

```python
def pow(base, exponent):
    return 1 if exponent == 0 else base * pow(base, exponent - 1)


def tests():
    assert solve() == math.factorial(100) - 2**100
    print("All tests passed")


def main():
    print()
    while True:
        print("1. Run tests")
        print("2. Use custom data")
        print("3. Exit")
        choice = int(input(": "))
        print("\n---\n")
        if choice == 1:
            tests()
        elif choice == 2:
            factorialNumber = int(input(f"Enter a number for the
factorial: "))
            print(
                "Below enter two numbers separated by space: power base
and the exponent: "
            )
            powerBase, powerExponent = list(map(int, input().split()))
            res = solve(factorialNumber, powerBase, powerExponent)
            print(f"\nResult: {res}")
        else:
            break
        print("\n---\n")


if __name__ == "__main__":
    main()
```

```python
"""
Обчислити 7^123
"""


def solve(powerBase=7, powerExponent=123):
    return pow(powerBase, powerExponent)


def pow(base, exponent):
    return 1 if exponent == 0 else base * pow(base, exponent - 1)
```

```python

def tests():
    assert solve() == 7**123
    print("All tests passed")


def main():
    print()
    while True:
        print("1. Run tests")
        print("2. Use custom data")
        print("3. Exit")
        choice = int(input(": "))
        print("\n---\n")
        if choice == 1:
            tests()
        elif choice == 2:
            print(f"Enter power base and power exponent below: ")
            powerBase, powerExponent = list(map(int, input().split()))
            res = solve(powerBase, powerExponent)
            print(f"\nResult: {res}")
        else:
            break
        print("\n---\n")


if __name__ == "__main__":
    main()
```

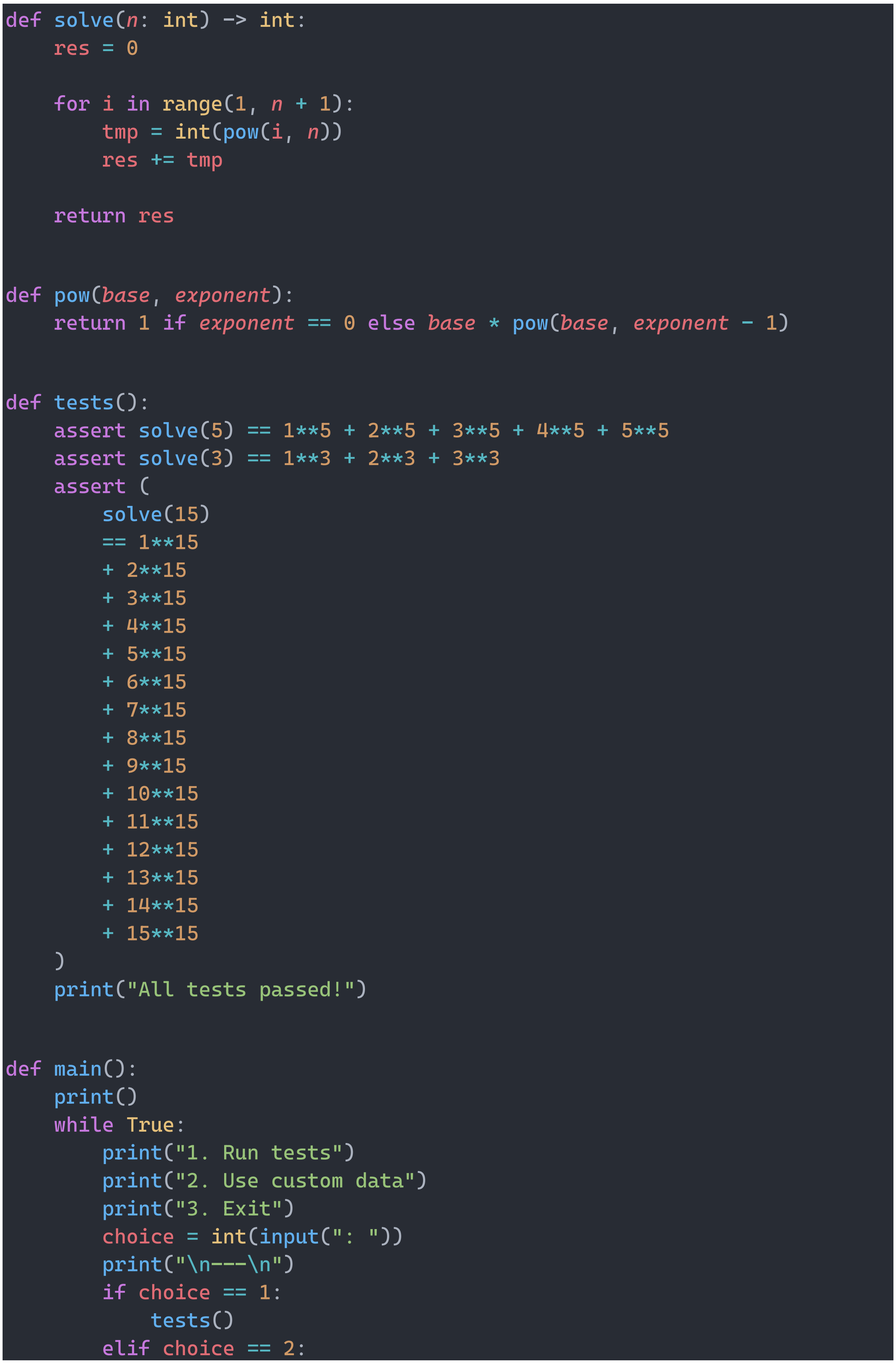
```python
"""
З'ясувати яке з чисел a**m чи b**n більше і на скільки. За умови, що a,b
<= 40000, a m,n <= 10
"""


def solve(a, b):
    biggerOne = a if a > b else b
    difference = b - a if b > a else a - b
    return biggerOne, difference


def pow(base, exponent):
    return 1 if exponent == 0 else base * pow(base, exponent - 1)


def tests():
    assert solve(pow(3, 2), pow(5, 3)) == (125, 116)
    assert solve(pow(7, 9), pow(9, 8)) == (43046721, 2693114)
    assert solve(pow(12, 3), pow(9, 16)) == (1853020188851841,
1853020188850113)
    assert solve(pow(24, 15), pow(17, 59)) == (
        394899297647654605580796211730554809533910274046242158741891554404
41816753,
        394899297647654605580796211730554809533910274046242108256163258799
5710129,
    )
    print("All tests passed!")


def main():
    print()
    while True:
        print("1. Run tests")
        print("2. Use custom data")
        print("3. Exit")
        choice = int(input(": "))
        print("\n---\n")
        if choice == 1:
            tests()
        elif choice == 2:
            print(
                "Enter two numbers for the first element, one for the
base and one for the exponent of the power, separated by a space below: "
            )
            oneBase, oneExponent = list(map(int, input().split()))
            one = pow(oneBase, oneExponent)

            print(
                "Enter two numbers for the second element, one for the
base of the power, another for the exponent of the power, separated by a
space below:"
            )
            twoBase, twoExponent = list(map(int, input().split()))
            two = pow(twoBase, twoExponent)

            bigger, difference = solve(one, two)
            print(f"\nResult: Larger one - {bigger}, difference -
{difference}")
        else:
            break
        print("\n---\n")


if __name__ == "__main__":
    main()
```

```python
"""
Скласти програму обчислення точного значення суми 1! + 2! + 3! + ⋯ + 𝑛!
за умовою, що 𝑛 > 10
"""

import math


def solve(nums: [int]) -> int:
    res = 0

    for num in nums:
        calculatedFactorial = factorial(num)
        res += calculatedFactorial

    return res


def factorial(n):
    return 1 if n == 1 else n * factorial(n - 1)


def tests():
    assert solve([1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11]) == math.factorial(
        1
    ) + math.factorial(2) + math.factorial(3) + math.factorial(4) +
math.factorial(
        5
    ) + math.factorial(
        6
    ) + math.factorial(
        7
    ) + math.factorial(
        8
    ) + math.factorial(
        9
    ) + math.factorial(
        10
    ) + math.factorial(
        11
    ), "Test 1 failed"
    assert solve([12, 13, 14, 15]) == math.factorial(12) +
math.factorial(
        13
    ) + math.factorial(14) + math.factorial(15), "Test 2 failed"
    assert solve([16, 17, 18, 19, 20]) == math.factorial(16) +
math.factorial(
        17
    ) + math.factorial(18) + math.factorial(19) + math.factorial(20),
"Test 3 failed"
    assert solve([2, 3, 9, 5]) == math.factorial(2) + math.factorial(
        3
    ) + math.factorial(9) + math.factorial(5)
    print("All tests passed")


def main():
    print()
    while True:
        print("1. Run tests")
        print("2. Enter custom data")
        print("3. Exit")
        choice = input(": ")
        print("\n---\n")
        if choice == "1":
            tests()
        elif choice == "2":
            print(
                f"Enter the numbers, for which you'd like to calculate
the sum of their factorials, or a number, up to which calculate the sum,
below:"
            )
            nums = list(map(int, input().split()))
            res = solve(nums) if len(nums) != 1 else solve(range(1,
nums[0] + 1))
            print(f"\nResult: {res}")
        else:
            break
        print("\n---\n")


if __name__ == "__main__":
    main()
```

```
"""
Обчислити точне значення суми 1**2 + 2**2 + 3**2 + ... + n**2 при умові,
що n >= 400
"""


def solve(nums: [int]) -> int:
    res = 0

    for num in nums:
```

```python
        calculatedSquare = num * num
        res += calculatedSquare

    return res


def tests():
    assert solve([1, 2, 3, 4, 5]) == 1**2 + 2**2 + 3**2 + 4**2 + 5**2
    assert solve([3, 9, 5, 4, 6]) == 3**2 + 9**2 + 5**2 + 4**2 + 6**2
    assert solve([11, 64, 32, 16]) == 11**2 + 64**2 + 32**2 + 16**2
    print("All tests solved!")


def main():
    print()
    while True:
        print("1. Run tests")
        print("2. Use custom data")
        print("3. Exit")
        choice = int(input(": "))
        print("\n---\n")
        if choice == 1:
            tests()
        elif choice == 2:
            print(
                "Enter a list of numbers, that you want to calculate the
sum of their squares, or just a number, up to which to calculate the sum,
below:"
            )
            nums = list(map(int, input().split()))
            res = solve(nums) if len(nums) != 1 else solve(range(1,
nums[0] + 1))
            print(f"\nResult: {res}")
        else:
            break
        print("\n---\n")


if __name__ == "__main__":
    main()
```

```python
"""
Обчислити точне значення суми 1**n + 2**n + 3**n + ... + n**n за умови,
що n >= 10
"""
```



```python
def solve(n: int) -> int:
    res = 0

    for i in range(1, n + 1):
        tmp = int(pow(i, n))
        res += tmp

    return res


def pow(base, exponent):
    return 1 if exponent == 0 else base * pow(base, exponent - 1)


def tests():
    assert solve(5) == 1**5 + 2**5 + 3**5 + 4**5 + 5**5
    assert solve(3) == 1**3 + 2**3 + 3**3
    assert (
        solve(15)
        == 1**15
        + 2**15
        + 3**15
        + 4**15
        + 5**15
        + 6**15
        + 7**15
        + 8**15
        + 9**15
        + 10**15
        + 11**15
        + 12**15
        + 13**15
        + 14**15
        + 15**15
    )
    print("All tests passed!")


def main():
    print()
    while True:
        print("1. Run tests")
        print("2. Use custom data")
        print("3. Exit")
        choice = int(input(": "))
        print("\n---\n")
        if choice == 1:
            tests()
        elif choice == 2:
            print("Enter your N number below:")
            n = int(input(": "))
            res = solve(n)
            print(f"\nResult: {res}")
        else:
            break
        print("\n---\n")


if __name__ == "__main__":
    main()
```

Висновки

Таким чином, ми вивчили основні можливості та принципи роботи із довгою арифметикою.