# Contents

# 1 Basic Test Results

```
Sat Nov 14 23:37:41 IST 2020
Process Process-78:
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/ex3/tsviel/final/testdir/lib/autotest.py", line 74, in wrap
    res=target(*args, **kwargs)
  File "/tmp/bodek.KRNToq/intro2cs1/ex3/tsviel/final/testdir/lib/testrunners.py", line 39, in import_runner
    code,res = peel(runners, modulename, fname, args, kwargs)
  File "/tmp/bodek.KRNToq/intro2cs1/ex3/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/ex3/tsviel/final/testdir/lib/testrunners.py", line 17, in check_args
    code,res = peel(runners, modulename, fname, args, kwargs)
  File "/tmp/bodek.KRNToq/intro2cs1/ex3/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/ex3/tsviel/final/testdir/lib/testrunners.py", line 12, in base_runner
    return None,func(*args, **kwargs)
  File "/tmp/bodek.KRNToq/intro2cs1/ex3/tsviel/final/testdir/src/ex3.py", line 184, in primes_for_asafi
    if is_prime(list(num_range)[i]):
IndexError: list index out of range
Process Process-82:
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/ex3/tsviel/final/testdir/lib/autotest.py", line 74, in wrap
    res=target(*args, **kwargs)
  File "/tmp/bodek.KRNToq/intro2cs1/ex3/tsviel/final/testdir/lib/testrunners.py", line 39, in import_runner
    code,res = peel(runners, modulename, fname, args, kwargs)
  File "/tmp/bodek.KRNToq/intro2cs1/ex3/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/ex3/tsviel/final/testdir/lib/testrunners.py", line 17, in check_args
    code,res = peel(runners, modulename, fname, args, kwargs)
  File "/tmp/bodek.KRNToq/intro2cs1/ex3/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/ex3/tsviel/final/testdir/lib/testrunners.py", line 12, in base_runner
    return None,func(*args, **kwargs)
  File "/tmp/bodek.KRNToq/intro2cs1/ex3/tsviel/final/testdir/src/ex3.py", line 191, in sum_of_vectors
    vector_length = len(vec_lst[0])
IndexError: list index out of range
Sat Nov 14 23:37:41 IST 2020
Archive:  /tmp/bodek.KRNToq/intro2cs1/ex3/tsviel/final/submission
  inflating: src/ex3.py
  inflating: src/__MACOSX/._ex3.py
7 passed tests out of 7 in test set named 'presubmit'.
result_code    presubmit    7    1
12 passed tests out of 12 in test set named 'input'.
result_code    input    12    1
14 passed tests out of 14 in test set named 'inner'.
result_code    inner    14    1
14 passed tests out of 14 in test set named 'monotonicity'.
result_code    monotonicity    14    1
22 passed tests out of 22 in test set named 'inverse'.
result_code    inverse    22    1
--> BEGIN TEST INFORMATION
Test name: prime_t8
```

```
Module tested: ex3
Function call: primes_for_asafi(5000)
Expected return value: [2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59, 61, 67, 71, 73, 79, 83, 89, 97, 101,
More test options: {}
--> END TEST INFORMATION
**********************************************************************
********************     There is a problem:
********************     The test named 'prime_t8' failed.
**********************************************************************
Test did not complete, exited with exitcode 1.
This probably means your code caused an exception to be raised.
result_code    prime_t8    exception    1
8 passed tests out of 9 in test set named 'prime'.
result_code    prime    8    1
--> BEGIN TEST INFORMATION
Test name: vecsums_t3
Module tested: ex3
Function call: sum_of_vectors([])
Expected return value: None
More test options: {}
--> END TEST INFORMATION
**********************************************************************
********************     There is a problem:
********************     The test named 'vecsums_t3' failed.
**********************************************************************
Test did not complete, exited with exitcode 1.
This probably means your code caused an exception to be raised.
result_code    vecsums_t3    exception    1
8 passed tests out of 9 in test set named 'vecsums'.
result_code    vecsums    8    1
11 passed tests out of 11 in test set named 'orthogonal'.
result_code    orthogonal    11    1
TESTING COMPLETED
```

## 2 ex3.py

```python
STOP = ""
WAITING_FOR_INPUT = True
PASSIVE_PRODUCT = 1
PASSIVE_SUM = 0


def lst_sum(lst):
    """The function returns the sum of a list of numbers"""
    the_sum = 0
    for i in range(len(lst)):
        the_sum += lst[i]
    return the_sum


def input_list():
    """The Function recieves numbers from users input,
    and returns them in a list from the first to the last one with there sum"""
    lst = []

    while WAITING_FOR_INPUT:
        user_input = input()
        if user_input == STOP:
            break;
        else:
            lst.append(float(user_input))
    return lst + [lst_sum(lst)]


def iter_product(iterable):
    """
    Returns the product of an Iterable
    :param iterable: Any iterable dataset (set, tuple,
    list, etc..) of floats\int
    :return: The inner product of the iterable unit
    """
    output = PASSIVE_PRODUCT
    for num in iterable:
        output *= num
    return output


def inner_product(vec_1, vec_2):
    """
    :param vec_1: a list of numbers(vector)
    :param vec_2: a list of numbers(vector)
    :return: A number(Float) represnting The inner product
    """
    # Validate if the abs of the vectors are equal
    if len(vec_1) != len(vec_2):
        return None
    # Validate if that the lists are not empty
    if len(vec_2) == 0 or len(vec_1) == 0:
        return 0
    # Action
    output = []
    for iterable in zip(vec_1, vec_2):
        output.append(iter_product(iterable))
    return lst_sum(output)

```

```python

def monotonicity_up(sequence):
    """
    :param sequence: an Iterable of numbers(floats or ints)
    :return: True if given sequence is Monotonicity up, False if not
    """
    cache = []
    for item in zip(sequence[:], sequence[1:]):
        if item[0] <= item[1]:
            cache.append(True)
        else:
            cache.append(False)
    return iter_product(cache)


def monotonicity_up_abs(sequence):
    """
    :param sequence: an Iterable of numbers(floats or ints)
    :return: True if given sequence is Monotonicity very up, False if not
    """
    cache = []
    for item in zip(sequence[:], sequence[1:]):
        if item[0] < item[1]:
            cache.append(True)
        else:
            cache.append(False)
    return iter_product(cache)


def monotonicity_down(sequence):
    """
    :param sequence: an Iterable of numbers(floats or ints)
    :return: True if given sequence is Monotonicity down, False if not
    """
    cache = []
    for item in zip(sequence[:], sequence[1:]):
        if item[0] >= item[1]:
            cache.append(True)
        else:
            cache.append(False)
    return iter_product(cache)


def monotonicity_down_abs(sequence):
    """
    :param sequence: an Iterable of numbers(floats or ints)
    :return: True if given sequence is Monotonicity very down, False if not
    """
    cache = []
    for item in zip(sequence[:], sequence[1:]):
        if item[0] > item[1]:
            cache.append(True)
        else:
            cache.append(False)
    return iter_product(cache)


def sequence_monotonicity(sequence):
    """
    :param sequence: List of Integers
    :return: List of Booleans
    """
    output = []  # The output to store our boolean answers
    if not isinstance(sequence, type(None)):
        if (sequence == [] or len(sequence) == 1):
            return [True, True, True, True]
        else:
            if monotonicity_up(sequence):  # Monotonicity Up Case
```

```python
                output.append(True)
            else:
                output.append(False)
            if monotonicity_up_abs(sequence):  # Monotonicity Very up Case
                output.append(True)
            else:
                output.append(False)
            if monotonicity_down(sequence):  # Monotonicity Down Case
                output.append(True)
            else:
                output.append(False)
            if monotonicity_down_abs(sequence):  # Monotonicity Very Down Case
                output.append(True)
            else:
                output.append(False)
            return output  # Return answer for monotonicity




def monotonicity_inverse(def_bool):
    if def_bool == [True, True, False, False]:
        return [-4, -3, 2, 5]  # Motonocity VERY UP
    if def_bool == [True, False, False, False]:
        return [1, 2, 2, 3] # MONOTONICITY UP
    if def_bool == [False, False, True, True]:
        return [4.2, 3, 0, -2]  #MONOTONICITY DOWN
    if def_bool == [False, False, True, False]:
        return [4, 3, 3, 1]  # MOTONICITY VERY DOWN
    if def_bool == [True, True, True, True]:
        return None
    if def_bool == [True, False, True, False]:
        return [1, 1, 1, 1]
    if def_bool == [False, False, False, False]:
        return [1, 2, -5, -1]
    else:
        return None


def is_prime(d):
    """True if n is prime number, False if not"""
    i = 2
    while i < d:
        if d % i == 0:
            return False
        i += 1
    return True


def primes_for_asafi(n):
    """ Help Asafi to return the prime numbers from 1 to n"""
    num_range = range(1, 10000)
    primes = []
    for i in num_range:
        if len(primes) == n:
            break
        if is_prime(list(num_range)[i]):
            primes.append(list(num_range)[i])
    return primes


def sum_of_vectors(vec_lst):
    """Returns the sum of of vectors"""
    vector_length = len(vec_lst[0])
    sum_vector = []
    for coordinate in range(vector_length):
        sum_vector_coordinate = []
        for vector_index in range(len(vec_lst)):
            sum_vector_coordinate.append(vec_lst[vector_index][coordinate])
        sum_vector.append(lst_sum(sum_vector_coordinate))
    return sum_vector


def num_of_orthogonal(vectors):
    """
    :param vectors: A Matrix (2D List) of vectors of same length
    :return: The amount of orthogonal Vectors
    """
    count = 0
    for vector_a in range(len(vectors)):
        for vector_b in range(vector_a+1, len(vectors)):
            if inner_product(vectors[vector_a], vectors[vector_b]) == 0:
                count += 1
    return count
```