```python
import convex_hull
from pytest import raises
import pytest

class Point:
        def __init__(self, x, y):
                self.x = x
                self.y = y


# test cases for white box - control flow - function left index
def test_case_1_control_flow():
    points = []
    points.append(Point(5, 9))

    assert convex_hull.Left_index(points) == 0

def test_case_2_control_flow():
    points = []
    points.append(Point(6, 5))
    points.append(Point(3, 6))

    assert convex_hull.Left_index(points) == 1

def test_case_3_control_flow():
    points = []
    points.append(Point(4, 7))
    points.append(Point(5, 7))

    assert convex_hull.Left_index(points) == 0

def test_case_4_control_flow():
    points = []
    points.append(Point(5, 4))
    points.append(Point(5, 1))

    assert convex_hull.Left_index(points) == 0

def test_case_5_control_flow():
    points = []
    points.append(Point(6, 3))
    points.append(Point(6, 7))

    assert convex_hull.Left_index(points) == 1

def test_case_extra_loop_1_control_flow():
    points = []
    points.append(Point(6, 5))
    points.append(Point(3, 6))
    points.append(Point(3, 7))

    assert convex_hull.Left_index(points) == 2

def test_case_extra_loop_2_control_flow():
    points = []
    points.append(Point(4, 7))
    points.append(Point(5, 7))
    points.append(Point(3, 5))

    assert convex_hull.Left_index(points) == 2

def test_case_extra_loop_3_control_flow():
    points = []
    points.append(Point(4, 7))
    points.append(Point(4, 6))
    points.append(Point(5, 9))

    assert convex_hull.Left_index(points) == 0

def test_case_extra_loop_4_control_flow():
    points = []
    points.append(Point(7, 9))
    points.append(Point(5, 8))
    points.append(Point(5, 6))

    assert convex_hull.Left_index(points) == 1

def test_case_extra_loop_5_control_flow():
    points = []
    points.append(Point(3, 4))
    points.append(Point(3, 5))
    points.append(Point(2, 9))

    assert convex_hull.Left_index(points) == 2

# test cases for white box - data flow - function convex hull

def test_case_variavel_points_data_flow():

    points = []
    points.append(Point(6, 8))
    points.append(Point(3, 4))
    points.append(Point(8, 5))

    expected_points =[]
    expected_points.append(Point(3, 4))
    expected_points.append(Point(8, 5))
    expected_points.append(Point(6, 8))

    result = convex_hull.convexHull(points,3)

    assert result[0].x == expected_points[0].x and result[0].y ==
expected_points[0].y and result[1].x == expected_points[1].x and
result[1].y == expected_points[1].y and result[2].x ==
expected_points[2].x and result[2].y == expected_points[2].y

def test_case_variavel_n_data_flow():
    points = []
    points.append(Point(3, 5))
    points.append(Point(6, 8))
    points.append(Point(10, 5))
    points.append(Point(4, 7))

    expected_points =[]
    expected_points.append(Point(3, 5))
    expected_points.append(Point(10, 5))
    expected_points.append(Point(6, 8))
    expected_points.append(Point(4, 7))

    result = convex_hull.convexHull(points,4)

    assert result[0].x == expected_points[0].x and result[0].y ==
expected_points[0].y and result[1].x == expected_points[1].x and
result[1].y == expected_points[1].y and result[2].x ==
expected_points[2].x and result[2].y == expected_points[2].y and
result[3].x == expected_points[3].x and result[3].y ==
expected_points[3].y

def test_case_variavel_l_data_flow():
    points = []
    points.append(Point(3, 5))
    points.append(Point(6, 8))
    points.append(Point(10, 5))
    points.append(Point(6, 6))

    expected_points =[]
    expected_points.append(Point(3, 5))
    expected_points.append(Point(10, 5))
    expected_points.append(Point(6, 8))

    result = convex_hull.convexHull(points,4)

    assert result[0].x == expected_points[0].x and result[0].y ==
expected_points[0].y and result[1].x == expected_points[1].x and
result[1].y == expected_points[1].y and result[2].x ==
expected_points[2].x and result[2].y == expected_points[2].y


def test_case_variavel_hull_data_flow():
    points = []
    points.append(Point(3, 6))
    points.append(Point(4, 8))
    points.append(Point(10, 5))
    points.append(Point(6, 7))

    expected_points =[]
    expected_points.append(Point(3, 6))
    expected_points.append(Point(10, 5))
    expected_points.append(Point(6, 7))
    expected_points.append(Point(4, 8))

    result = convex_hull.convexHull(points,4)

    assert result[0].x == expected_points[0].x and result[0].y ==
expected_points[0].y and result[1].x == expected_points[1].x and
result[1].y == expected_points[1].y and result[2].x ==
expected_points[2].x and result[2].y == expected_points[2].y and
result[3].x == expected_points[3].x and result[3].y ==
expected_points[3].y

def test_case_variavel_p_data_flow():
    points = []
    points.append(Point(3, 6))
    points.append(Point(4, 8))
    points.append(Point(10, 5))
    points.append(Point(6, 7))
    points.append(Point(6, 8))

    expected_points =[]
    expected_points.append(Point(3, 6))
    expected_points.append(Point(10, 5))
    expected_points.append(Point(6, 8))
    expected_points.append(Point(4, 8))

    result = convex_hull.convexHull(points,5)

    assert result[0].x == expected_points[0].x and result[0].y ==
expected_points[0].y and result[1].x == expected_points[1].x and
result[1].y == expected_points[1].y and result[2].x ==
expected_points[2].x and result[2].y == expected_points[2].y and
result[3].x == expected_points[3].x and result[3].y ==
expected_points[3].y

def test_case_variavel_q_data_flow():
    points = []
    points.append(Point(4, 8))
    points.append(Point(10, 5))
    points.append(Point(6, 7))
    points.append(Point(6, 8))

    expected_points =[]
    expected_points.append(Point(4, 8))
    expected_points.append(Point(10, 5))
    expected_points.append(Point(6, 8))

    result = convex_hull.convexHull(points,4)

    assert result[0].x == expected_points[0].x and result[0].y ==
expected_points[0].y and result[1].x == expected_points[1].x and
result[1].y == expected_points[1].y and result[2].x ==
expected_points[2].x and result[2].y == expected_points[2].y


def test_case_variavel_each_data_flow():
    points = []
    points.append(Point(6, 8))
    points.append(Point(1, 6))
    points.append(Point(4, 8))
    points.append(Point(6, 7))

    expected_points =[]
    expected_points.append(Point(1, 6))
    expected_points.append(Point(6, 7))
    expected_points.append(Point(6, 8))
    expected_points.append(Point(4, 8))

    result = convex_hull.convexHull(points,4)
    assert result[0].x == expected_points[0].x and result[0].y ==
expected_points[0].y and result[1].x == expected_points[1].x and
result[1].y == expected_points[1].y and result[2].x ==
expected_points[2].x and result[2].y == expected_points[2].y and
result[3].x == expected_points[3].x and result[3].y ==
expected_points[3].y



# test case - black box testing, invalid and valid equivalence
classes

def test_left_index_valid_classes():
    points = []
    points.append(Point(5, 9))

    assert convex_hull.Left_index(points) == 0

    points = []
    points.append(Point(4, 7))
    points.append(Point(5, 7))
    points.append(Point(3, 5))

    assert convex_hull.Left_index(points) == 2

    points = []
    points.append(Point(float(4.00), float(7.00)))
    points.append(Point(float(5.00), float(7.00)))
    points.append(Point(float(3.00), float(5.00)))

    assert convex_hull.Left_index(points) == 2

    points = []
    points.append(Point("4", "7"))
    points.append(Point("5", "7"))
    points.append(Point("3", "5"))
    assert convex_hull.Left_index(points) == 2

    points = []
    points.append(Point(float(4.000000001), float(7.000000001)))
    points.append(Point(float(5.000000001), float(7.000000001)))
    points.append(Point(float(3.000000001), float(5.000000001)))

    assert convex_hull.Left_index(points) == 2

    points = []
    points.append(Point(-4, -7))
    points.append(Point(-5,- 7))
    points.append(Point(-3, -5))

    assert convex_hull.Left_index(points) == 1

    points = []
    points.append(Point(float(-4.00), float(-7.00)))
    points.append(Point(float(-5.00), float(-7.00)))
    points.append(Point(float(-3.00), float(-5.00)))
```

```python
    assert convex_hull.Left_index(points) == 1

    points = []
    points.append(Point("4.00", "7.00"))
    points.append(Point("5.00", "7.00"))
    points.append(Point("3.00", "5.00"))
    assert convex_hull.Left_index(points) == 2

    points = []
    points.append(Point(float(4.000000001),  "7.00"))
    points.append(Point(float(5.00), 7))
    points.append(Point(3.0, "5.00"))
    assert convex_hull.Left_index(points) == 2

    points = []
    points.append(Point(True, True))
    points.append(Point(False, False))
    points.append(Point(True, False))
    assert convex_hull.Left_index(points) == 1

    points = []
    points.append(Point(3+1, 5+2))
    points.append(Point(3+2, 5+2))
    points.append(Point(2+1, 4+1))
    assert convex_hull.Left_index(points) == 2

    points = []
    assert convex_hull.Left_index(points) == 0


def test_left_index_invalid_classes():

    points = []
    points.append(Point(float(4.000000001),  "7.00"))
    points.append(Point(float(5.00), 7))
    points.append(Point("3.0", "5.00"))
    with raises(TypeError):  #cant compare string with float
        assert convex_hull.Left_index(points) == 2

    points = []
    points.append(Point("-4", "-7"))
    points.append(Point("-5", "-7"))
    points.append(Point("-3", "-5"))
    with raises(AssertionError): #cant convert '-' to represent a 
negative value
        assert convex_hull.Left_index(points) == 1

    points = []
    points.append((4, 7))
    points.append((5, 7))
    points.append((3, 5))
    with raises(AttributeError): #list of points is not a classe 
Point list
        assert convex_hull.Left_index(points) == 2

    points = []
    points.append(("abc", "too"))
    points.append(("ma", "do"))
    points.append(("ti", "ai"))
    with raises(AttributeError): #list of points is not a classe
Point list
        assert convex_hull.Left_index(points) == 2

    points = []
    points.append((True, True))
    points.append((False, False))
    points.append((True, False))
    with raises(AttributeError): #list of points is not a classe
Point list
        assert convex_hull.Left_index(points) == 1

    points = True
    with raises(TypeError): #bool does not have len()
        assert convex_hull.Left_index(points) == 2

    points = "abc"
    with raises(AttributeError): #list of points is not a classe
Point list
        assert convex_hull.Left_index(points) == 2

    points = {(6,5),(7,8)}
    with raises(TypeError): #list of points is not a classe Point
list
        assert convex_hull.Left_index(points) == 2


def test_orientation_valid_classes():

    p=Point(4, 7)
    q=Point(5, 7)
    r=Point(3, 5)
    assert convex_hull.orientation(p,q,r) == 1


    p=Point(3, 5)
    q=Point(5, 7)
    r=Point(4, 7)
    assert convex_hull.orientation(p,q,r) == 2

    p=Point(3, 5)
    q=Point(3, 6)
    r=Point(3, 7)
    assert convex_hull.orientation(p,q,r) == 0


    p= Point(float(4.00), float(7.00))
    q= Point(float(5.00), float(7.00))
    r=Point(float(3.00), float(5.00))
    assert convex_hull.orientation(p,q,r) == 1


    p=Point(float(4.000000001), float(7.000000001))
    r=Point(float(5.000000001), float(7.000000001))
    q=Point(float(3.000000001), float(5.000000001))
    assert convex_hull.orientation(p,q,r) == 2


    p=Point(-4, -7)
    r=Point(-5,- 7)
    q=Point(-3, -5)

    assert convex_hull.orientation(p,q,r) == 2


    p=Point(float(-4.00), float(-7.00))
    r=Point(float(-5.00), float(-7.00))
    q=Point(float(-3.00), float(-5.00))

    assert convex_hull.orientation(p,q,r) == 2


    p=Point(float(4.000000001), float(7.00))
    r=Point(float(5.00), 7)
    q=Point(3.0, float(5.00))
    assert convex_hull.orientation(p,q,r) == 2


    p=Point(True, True)
    r=Point(False, False)
    q=Point(True, False)
    assert convex_hull.orientation(p,q,r) == 1


    p=Point(3+1, 5+2)
    r=Point(3+2, 5+2)
    q=Point(2+1, 4+1)
    assert convex_hull.orientation(p,q,r) == 2


def test_orientation_invalid_classes():

    p=Point("4", "7")
    q=Point("5", "7")
    r=Point("3", "5")
    with raises(TypeError): #cant do math operations with strings
        assert convex_hull.orientation(p,q,r) == 1


    p=Point(float(4.000000001), "7")
    r=Point(float(5.00), 7)
    q=Point(3.0, "3")
    with raises(TypeError):# cant compare strings with floats
        assert convex_hull.orientation(p,q,r) == 2


    p=(3, 5)
    q=(3, 6)
    r=(3, 7)
    with raises(AttributeError): #list of points is not a classe
Point list
```

```
        assert convex_hull.orientation(p,q,r) == 0


    p=[3, 5]
    q=[3, 6]
    r=[3, 7]
    with raises(AttributeError): #list of points is not a classe
Point list
        assert convex_hull.orientation(p,q,r) == 1


    p=4
    r=7
    q=8
    with raises(AttributeError): #list of points is not a classe
Point list
        assert convex_hull.orientation(p,q,r) == 2


    p=(True, True)
    r=(False, False)
    q=(True, False)
    with raises(AttributeError): #list of points is not a classe
Point list
        assert convex_hull.orientation(p,q,r) == 2


def test_convexHull_valid_classes():
    points = []
    points.append(Point(5, 9))

    assert convex_hull.convexHull(points,1) == None

    points = []
    points.append(Point(4, 7))
    points.append(Point(5, 7))
    points.append(Point(3, 5))

    expected_points =[]
    expected_points.append(Point(3, 5))
    expected_points.append(Point(5, 7))
    expected_points.append(Point(4, 7))

    result = convex_hull.convexHull(points,3)

    assert result[0].x == expected_points[0].x and result[0].y ==
expected_points[0].y and result[1].x == expected_points[1].x and
result[1].y == expected_points[1].y and result[2].x ==
expected_points[2].x and result[2].y == expected_points[2].y


    points = []
    points.append(Point(float(4.00), float(7.00)))
    points.append(Point(float(5.00), float(7.00)))
    points.append(Point(float(3.00), float(5.00)))

    expected_points =[]
    expected_points.append(Point(3, 5))
    expected_points.append(Point(5, 7))
    expected_points.append(Point(4, 7))

    result = convex_hull.convexHull(points,3)

    assert result[0].x == expected_points[0].x and result[0].y ==
expected_points[0].y and result[1].x == expected_points[1].x and
result[1].y == expected_points[1].y and result[2].x ==
expected_points[2].x and result[2].y == expected_points[2].y


    points = []
    points.append(Point(float(4.000000001), float(7.000000001)))
    points.append(Point(float(5.000000001), float(7.000000001)))
    points.append(Point(float(3.000000001), float(5.000000001)))

    expected_points =[]
    expected_points.append(Point(float(3.000000001),
float(5.000000001)))
    expected_points.append(Point(float(5.000000001),
float(7.000000001)))
    expected_points.append(Point(float(4.000000001),
float(7.000000001)))

    result = convex_hull.convexHull(points,3)

    assert result[0].x == expected_points[0].x and result[0].y ==
expected_points[0].y and result[1].x == expected_points[1].x and
result[1].y == expected_points[1].y and result[2].x ==
expected_points[2].x and result[2].y == expected_points[2].y


    points = []
    points.append(Point(-4, -7))
    points.append(Point(-5,- 7))
    points.append(Point(-3, -5))

    expected_points =[]
    expected_points.append(Point(-5,- 7))
    expected_points.append(Point(-4, -7))
    expected_points.append(Point(-3, -5))

    result = convex_hull.convexHull(points,3)

    assert result[0].x == expected_points[0].x and result[0].y ==
expected_points[0].y and result[1].x == expected_points[1].x and
result[1].y == expected_points[1].y and result[2].x ==
expected_points[2].x and result[2].y == expected_points[2].y


    points = []
    points.append(Point(float(-4.00), float(-7.00)))
    points.append(Point(float(-5.00), float(-7.00)))
    points.append(Point(float(-3.00), float(-5.00)))
    expected_points =[]
    expected_points.append(Point(float(-5.00), float(-7.00)))
    expected_points.append(Point(float(-4.00), float(-7.00)))
    expected_points.append(Point(float(-3.00), float(-5.00)))

    result = convex_hull.convexHull(points,3)

    assert result[0].x == expected_points[0].x and result[0].y ==
expected_points[0].y and result[1].x == expected_points[1].x and
result[1].y == expected_points[1].y and result[2].x ==
expected_points[2].x and result[2].y == expected_points[2].y


    points = []
    points.append(Point(float(4.000000001),  float(7.00)))
    points.append(Point(float(5.00), 7))
    points.append(Point(3,float(5.00)))

    expected_points =[]
    expected_points.append(Point(3, float(5.00)))
    expected_points.append(Point(float(5.00), 7))
    expected_points.append(Point(float(4.000000001), float(7.00)))

    result = convex_hull.convexHull(points,3)

    assert result[0].x == expected_points[0].x and result[0].y ==
expected_points[0].y and result[1].x == expected_points[1].x and
result[1].y == expected_points[1].y and result[2].x ==
expected_points[2].x and result[2].y == expected_points[2].y

    points = []
    points.append(Point(True, True))
    points.append(Point(False, False))
    points.append(Point(True, False))

    expected_points =[]
    expected_points.append(Point(False, False))
    expected_points.append(Point(True, False))
    expected_points.append(Point(True, True))

    result = convex_hull.convexHull(points,3)

    assert result[0].x == expected_points[0].x and result[0].y ==
expected_points[0].y and result[1].x == expected_points[1].x and
result[1].y == expected_points[1].y and result[2].x ==
expected_points[2].x and result[2].y == expected_points[2].y

    points = []
    points.append(Point(3+1, 5+2))
    points.append(Point(3+2, 5+2))
    points.append(Point(2+1, 4+1))

    expected_points =[]
    expected_points.append(Point(3, 5))
    expected_points.append(Point(5, 7))
    expected_points.append(Point(4, 7))
    result = convex_hull.convexHull(points,3)

    assert result[0].x == expected_points[0].x and result[0].y ==
expected_points[0].y and result[1].x == expected_points[1].x and
result[1].y == expected_points[1].y and result[2].x ==
expected_points[2].x and result[2].y == expected_points[2].y

    points = []
    assert convex_hull.convexHull(points,0) == None


def test_convexHull_invalid_classes():

    points = []
    points.append(Point(4, 7))
    points.append(Point(5, 7))
    points.append(Point(3, 5))

    expected_points =[]
    expected_points.append(Point (3, 5))
    expected_points.append(Point (5, 7))
    expected_points.append(Point (4, 7))

    with raises(TypeError): #cant compare strings with integers
        result = convex_hull.convexHull(points,"3")
        assert result[0].x == expected_points[0].x and result[0].y
== expected_points[0].y and result[1].x == expected_points[1].x and
result[1].y == expected_points[1].y and result[2].x ==
expected_points[2].x and result[2].y == expected_points[2].y


    points = []
    points.append(Point(4, 7))
    points.append(Point(5, 7))
    points.append(Point(3, 5))

    expected_points =[]
    expected_points.append(Point (3, 5))
    expected_points.append(Point (5, 7))
    expected_points.append(Point (4, 7))

    with raises(TypeError): #float cannot be interpreted as integer
        result = convex_hull.convexHull(points,float(3.0))
        assert result[0].x == expected_points[0].x and result[0].y
== expected_points[0].y and result[1].x == expected_points[1].x and
result[1].y == expected_points[1].y and result[2].x ==
expected_points[2].x and result[2].y == expected_points[2].y


    points = []
    points.append(Point(float(4.000000001),  "7.00"))
    points.append(Point(float(5.00), 7))
    points.append(Point("3.0", "5.00"))

    expected_points =[]
    expected_points.append(Point (float (4.000000001),  "7.00"))
    expected_points.append(Point (float (5.00), 7))
    expected_points.append(Point("3.0", "5.00"))

    with raises(TypeError):  #cant compare string with float
        result = convex_hull.convexHull(points,3)
        assert result[0].x == expected_points[0].x and result[0].y
== expected_points[0].y and result[1].x == expected_points[1].x and
result[1].y == expected_points[1].y and result[2].x ==
expected_points[2].x and result[2].y == expected_points[2].y


    points = []
    points.append(Point("-4", "-7"))
    points.append(Point("-5", "-7"))
    points.append(Point("-3", "-5"))

    expected_points =[]
    expected_points.append(Point("-5", "-7"))
    expected_points.append(Point("-4", "-7"))
    expected_points.append(Point("-3", "-5"))

    with raises(TypeError): #cant do math operations with strings
        result = convex_hull.convexHull(points,3)
        assert result[0].x == expected_points[0].x and result[0].y
== expected_points[0].y and result[1].x == expected_points[1].x and
result[1].y == expected_points[1].y and result[2].x ==
expected_points[2].x and result[2].y == expected_points[2].y


    points = []
    points.append((4, 7))
    points.append((5, 7))
    points.append((3, 5))

    expected_points =[]
    expected_points.append((3, 5))
    expected_points.append((5, 7))
    expected_points.append((4, 7))


    with raises(AttributeError):  #list of points is not a classe
Point list
        result = convex_hull.convexHull(points,3)
        assert result[0].x == expected_points[0].x and result[0].y
== expected_points[0].y and result[1].x == expected_points[1].x and
result[1].y == expected_points[1].y and result[2].x ==
expected_points[2].x and result[2].y == expected_points[2].y


    points = []
    points.append(("abc", "too"))
    points.append(("ma", "do"))
    points.append(("ti", "ai"))

    expected_points =[]
    expected_points.append(("abc", "too"))
    expected_points.append(("ma", "do"))
    expected_points.append(("ti", "ai"))
    with raises(AttributeError):  #list of points is not a classe
Point list
        result = convex_hull.convexHull(points,3)
        assert result[0].x == expected_points[0].x and result[0].y
== expected_points[0].y and result[1].x == expected_points[1].x and
result[1].y == expected_points[1].y and result[2].x ==
expected_points[2].x and result[2].y == expected_points[2].y

    points = []
    points.append((True, True))
    points.append((False, False))
    points.append((True, False))

    expected_points =[]
    expected_points.append((True, True))
    expected_points.append((False, False))
    expected_points.append((True, False))

    with raises(AttributeError):  #list of points is not a classe
Point list
        result = convex_hull.convexHull(points,3)
        assert result[0].x == expected_points[0].x and result[0].y
== expected_points[0].y and result[1].x == expected_points[1].x and
result[1].y == expected_points[1].y and result[2].x ==
expected_points[2].x and result[2].y == expected_points[2].y

# test case - black box testing, boundary values
def test_boundary_convex_hull():

    points = []
    assert convex_hull.convexHull(points,0) == None

    points = []
    points.append(Point(5, 9))

    assert convex_hull.convexHull(points,1) == None

    points = []
    points.append(Point(4, 7))
    points.append(Point(5, 7))
    points.append(Point(3, 5))

    expected_points =[]
    expected_points.append(Point (3, 5))
    expected_points.append(Point (5, 7))
    expected_points.append(Point (4, 7))

    result = convex_hull.convexHull(points,3)

    assert result[0].x == expected_points[0].x and result[0].y ==
expected_points[0].y and result[1].x == expected_points[1].x and
result[1].y == expected_points[1].y and result[2].x ==
expected_points[2].x and result[2].y == expected_points[2].y
```

```
    points = []
    points.append(Point(5, 9))
    with raises(ValueError):
        assert convex_hull.convexHull(points,-1) == None
```