```
#!/usr/bin/env python3

import re, sys

try:
    # if the height is even
    if int(sys.argv[1]) % 2 == 0:
        height = int(sys.argv[1]) + 1
    else:
        height = int(sys.argv[1])
    # if the length is even
    if int(sys.argv[2]) % 2 == 0:
        length = int(sys.argv[2]) + 1
    else:
        length = int(sys.argv[2])
except:
    print("Error: height and length must be positive integers.\n\
Usage: <simwalk.py> <height> <length> <input file>")
    sys.exit()

# the town boundaries.
north_wall = int(height / 2) # north wall
south_wall = -1*int(height / 2) # south wall
west_wall = -1*int(length / 2) # west wall
east_wall = int(length / 2) # east wall

# the origin point (0, 0), which also is the initial position.
x = 0
y = 0
# how many times it returns to the origin.
return_to_origin = 0
# number of steps
steps = 0

if int(sys.argv[1]) <= 0 or int(sys.argv[2]) <= 0: # if height and
length are negtive numbers.
    print("Error: height and length must be positive.\n\
Usage: <simwalk.py> <height> <length> <walk>")
    sys.exit()

if len(sys.argv) != 4: # if the number of arguments not equals to 3
    print("Error: simwalk requires exactly 3 arguments.\n\
Usage: <simwalk.py> <height> <length> <walk")
    sys.exit()


if int(sys.argv[1]) < 10 or int(sys.argv[2]) < 10: # if height and
length are smaller than 10.
    print("Error: height and length may not be smaller than 10.\n\
Usage: <simwalk.py> <height> <length> <walk>")
```

```
    sys.exit()


try:
    with open(sys.argv[3]) as walk: # read walk file.
        file = walk.read()

        # if white-space chars in the walk file.
        file = ''.join(file.split())

        match = re.findall("[^nesw]+", file)
        # if there are chars which are not 'n e s w'.
        if match:
            print("The input file has corrupt data.")
            sys.exit()

        for line in file:
            # define the ending reason.
            reason = ''

            for char in line:
                # add one step.
                steps += 1

                if char == 'n':
                    # move towards to north.
                    y += 1

                    # if return to the origin (0, 0).
                    if x == 0 and y == 0:
                        return_to_origin += 1

                    # if it reaches to the north wall.
                    if y >= north_wall:
                        reason = 'the north wall was reached.'
                        break

                if char == 's':
                    # move towards to south.
                    y -= 1

                    # if return to the origin (0, 0).
                    if x == 0 and y == 0:
                        return_to_origin += 1

                    # if it reaches to the south wall.
                    if y <= south_wall:
                        reason = 'the south wall was reached.'
                        break
                if char == 'e':
                    # move towards to east.
                    x += 1

                    # if return to the origin (0, 0).
                    if x == 0 and y == 0:
                        return_to_origin += 1

                    # if it reaches to the east wall.
                    if x >= east_wall:

                        # if the gates were reached.
                        if -2 <= y <= 2:
                            reason = 'the gate was reached.'
                            break
                        else:
                            # the east walls were reached.
                            reason = 'the east wall was reached.'
                            break

                if char == 'w':
                    # move towards to west.
                    x -= 1

                    # if return to the origin (0, 0).
                    if x == 0 and y == 0:
                        return_to_origin += 1

                    # if it reaches to the west wall.
                    if x <= west_wall:
                        reason = 'the west wall was reached.'
                        break

            if reason:
                walk.close()
                break

        walk.close()
        if steps == len(file):
        # if the entire file has been read.
            reason = "the sequence of steps has ended."
        end = "The walk was {} steps long. Returned to origin {} \
time(s). It ended because {}".format(steps, return_to_origin, reason)
        print(end)

except PermissionError:
    print('User doesnt have incorrect file permissions.')
    sys.exit()

except FileNotFoundError:
```

```
print('File is not found.')
sys.exit()
```