1.4.28

BEERAM MADHURI - EE25BTECH11012

August 2025

Question

Find the position vector of a point **R** which divides the line joining two points **P** and **Q** whose position vectors are $(2\mathbf{a} + \mathbf{b})$ and $(\mathbf{a} - 3\mathbf{b})$ externally in the ratio $1 : 2$. Also, show that **P** is the mid point of the line segment RQ.

given data

R divides **p** and **Q** in 1:2 ratio.

variable	Position Vector
P	$2a + b$
Q	$a - 3b$

Formula

Section Formula for a vector **R** which divides **P** and **Q** in k:1 ratio externally is given by

$$\mathbf{R} = \frac{k(\mathbf{P}) - 1(\mathbf{Q})}{k - 1}$$

finding Position vector of **R**

$$\mathbf{R} = \frac{2(\mathbf{P}) - 1(\mathbf{Q})}{2-1} \tag{1}$$

$$= \frac{2(2a+b) - (a-3b)}{1} \tag{2}$$

$$= 3a + 5b \tag{3}$$

$$\tag{4}$$

Hence Position vector of **R** is 3a + 5b

Proving **P** is midpoint of **QR**

$$\mathbf{P} = \frac{k(\mathbf{R}) + 1(\mathbf{Q})}{k+1}$$

$$2a + b = \frac{k(3a + 5b) + a - 3b}{k+1}$$

$$(2a + b)(k+1) = (3k+1)a + (5k-3)b$$

Comparing coefficients of a:

$$2k + 2 = 3k + 1$$

$$k = 1$$

Hence **P** divides $\overline{RQ}$ in 1:1 ratio, P is midpoint of $\overline{RQ}$.

Python Code

```
 import matplotlib.pyplot as plt
import numpy as np

# Define Base Vectors
# We assign arbitrary coordinates to vectors 'a' and 'b' for
    visualization.
# To see a different layout, you can change these values.
a = np.array([1, 1])
b = np.array([-1, 2])
```

Python Code

```
# Define Position Vectors for P and Q
# As given in the problem statement.
P = 2*a + b
Q = a - 3*b
 # Calculate Position Vector for R ---
# Using the external division formula result we found: R = 3a + 5
    b
R = 3*a + 5*b
```

Python Code

```
# Verify P is the midpoint of RQ ---
# This calculation should result in the same coordinates as P.
midpoint_RQ = (R + Q) / 2
print(f"Coordinates of P: {P}")
print(f"Calculated midpoint of RQ: {midpoint_RQ}")
print(f"Is P the midpoint of RQ? {np.allclose(P, midpoint_RQ)}")
```

Python Code

```
# --- Create the Plot ---
fig, ax = plt.subplots(figsize=(10, 8))
# Plot the line segment RQ
ax.plot([R[0], Q[0]], [R[1], Q[1]], 'k--', alpha=0.6, label='Line
    Segment RQ')
# Plot the points O, P, Q, R
ax.scatter(0, 0, c='black', s=100, zorder=5, label='Origin (O)')
ax.scatter(P[0], P[1], c='red', s=100, zorder=5, label=f'P = {P}'
    )
ax.scatter(Q[0], Q[1], c='green', s=100, zorder=5, label=f'Q = {Q
    }')
ax.scatter(R[0], R[1], c='blue', s=100, zorder=5, label=f'R = {R}
    ')
```

Python Code

```
# Plot position vectors from the origin
ax.quiver(0, 0, P[0], P[1], angles='xy', scale_units='xy', scale
    =1, color='red', alpha=0.7)
ax.quiver(0, 0, Q[0], Q[1], angles='xy', scale_units='xy', scale
    =1, color='green', alpha=0.7)
ax.quiver(0, 0, R[0], R[1], angles='xy', scale_units='xy', scale
    =1, color='blue', alpha=0.7)
```

Python Code

```
# Add Labels and Formatting
# Add text labels for each point
ax.text(0, 0.5, 'O', fontsize=14)
ax.text(P[0] + 0.3, P[1], 'P', fontsize=14)
ax.text(Q[0] + 0.3, Q[1], 'Q', fontsize=14)
ax.text(R[0] + 0.3, R[1], 'R', fontsize=14)
```

Python Code

```
# Set plot aesthetics
ax.set_title('Vector Visualization', fontsize=16)
ax.set_xlabel('X-axis', fontsize=12)
ax.set_ylabel('Y-axis', fontsize=12)
ax.axhline(0, color='grey', linewidth=0.5)
ax.axvline(0, color='grey', linewidth=0.5)
ax.grid(True, which='both', linestyle='--', linewidth=0.5)
ax.set_aspect('equal', adjustable='box')
ax.legend()

plt.show()
```

C Code

```c
#include <stdio.h>

// A structure to represent a vector with coefficients for a and
    b
typedef struct {
    int coeff_a;
    int coeff_b;
} Vector;
int main() {
    // P = 2a + 1b
    Vector P = {2, 1};
```

C Code

```
// Q = 1a - 3b
Vector Q = {1, -3};
// Ratio m:n = 1:2
int m = 1;
int n = 2;
// Applying the external division formula: R = (m*Q - n*P) /
    (m - n)
// Numerator: (1 * Q) - (2 * P)
int num_a = (m * Q.coeff_a) - (n * P.coeff_a); // (1*1) -
    (2*2) = -3
int num_b = (m * Q.coeff_b) - (n * P.coeff_b); // (1*-3) -
    (2*1) = -5
```

C Code

```
    // Denominator: m - n
    int den = m - n; // 1 - 2 = -1
    Vector R;
    R.coeff_a = num_a / den; // -3 / -1 = 3
    R.coeff_b = num_b / den; // -5 / -1 = 5

    printf("The position vector of point R is (%d)a + (%d)b\n", R
        .coeff_a, R.coeff_b);
return 0;
}
```

Python and C Code

```
# File: main.py
import ctypes
import platform

# --- 1. Define a Python class that mirrors the C struct ---
# The names and types in fields must exactly match the C struct.
class Vector(ctypes.Structure):
    fields = [("coeff_a", ctypes.c_int),
              ("coeff_b", ctypes.c_int)]
```

Python and C Code

```
# --- 2. Load the shared library ---
if platform.system() == "Windows":
    lib_path = "./libvector.dll"
else:
    lib_path = "./libvector.so"

try:
    lib = ctypes.CDLL(lib_path)
except OSError as e:
    print(f"Error loading library: {e}")
    print("Have you compiled vector_ops.c?")
    exit()
```

Python and C Code

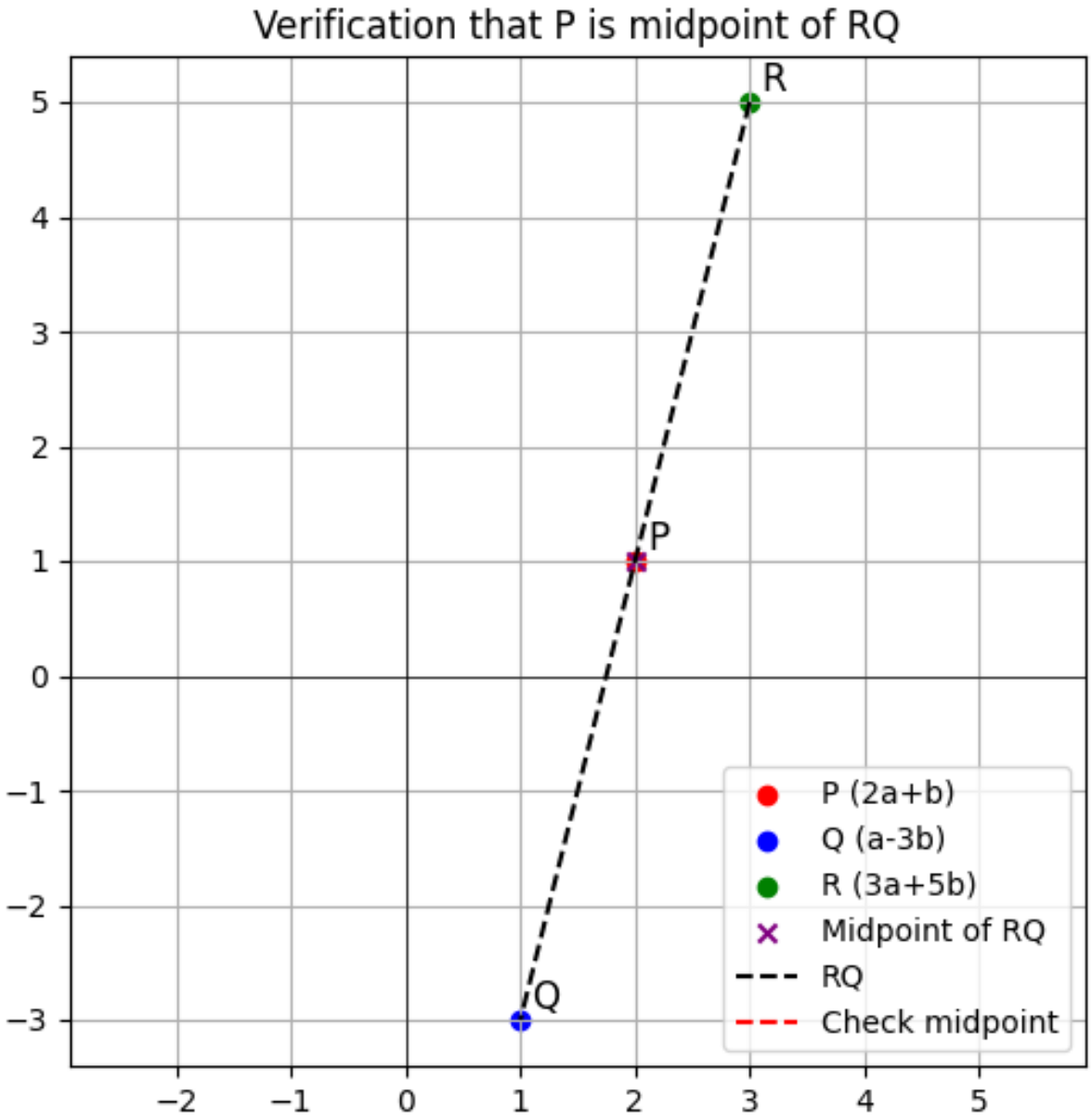
```
# --- 3. Define the function signature (argtypes and restype) ---
# This ensures Python sends the correct data types to the C
    function.
lib.external_division.argtypes = [
    ctypes.POINTER(Vector), # const Vector* P
    ctypes.POINTER(Vector), # const Vector* Q
    ctypes.c_int, # int m
    ctypes.c_int, # int n
    ctypes.POINTER(Vector) # Vector* R (output)
]
lib.external_division.restype = None # for void return type
```

Python and C Code

```
# --- 4. Prepare Input and Output Data ---
# Create instances of our Vector class for the inputs
P = Vector(coeff_a=2, coeff_b=1)
Q = Vector(coeff_a=1, coeff_b=-3)
# Define the ratio m:n
m = 1
n = 2
# Create an empty Vector instance to hold the result from the C
    function.
# This acts as the output buffer.
R = Vector()
```

Python and C Code

```
# --- 5. Call the C function ---
# The C function will write its result directly into our 'R'
    object.
lib.external_division(ctypes.byref(P), ctypes.byref(Q), m, n,
    ctypes.byref(R))
print(f"Vector P = ({P.coeff_a})a + ({P.coeff_b})b")
print(f"Vector Q = ({Q.coeff_a})a + ({Q.coeff_b})b")
print(f"Ratio m:n = {m}:{n}")
print("-" * 30)
print(f"The position vector of point R is ({R.coeff_a})a + ({R.
    coeff_b})b")
```


Verification that P is midpoint of RQ
R
P
Q
P (2a+b)
Q (a-3b)
R (3a+5b)
Midpoint of RQ
RQ
Check midpoint
5
4
3
2
1
0
−1
−2
−3
−2
−1
0
1
2
3
4
5