# 1.5.36

Sai Krishna Bakki - EE25BTECH11049

# Question

Point P$(x, 4)$ lies on the line segment joining the points **A**$(-5, 8)$ and **B**$(4, -10)$. Find the ratio in which point P divides the line segment AB. Also, find the value of x.

# Theoretical Solution

Let

$$\mathbf{A} = \begin{pmatrix} -5 \\ 8 \end{pmatrix}, \mathbf{B} = \begin{pmatrix} 4 \\ -10 \end{pmatrix}$$

Let $\mathbf{P} = \lambda\mathbf{A} + \mu\mathbf{B}$ with $\lambda + \mu = 1$. Using the y-coordinates:

$$\begin{pmatrix} 8 & -10 \\ 1 & 1 \end{pmatrix} \begin{pmatrix} \lambda \\ \mu \end{pmatrix} = \begin{pmatrix} 4 \\ 1 \end{pmatrix} \tag{1}$$

Hence the internal division ratio

$$AP : PB = \mu : \lambda = 2 : 7 \tag{2}$$

and

$$x = \lambda(-5) + \mu(4) = -\frac{35}{9} + \frac{8}{9} = -3 \tag{3}$$

So, $\mathbf{P} = (-3, 4)$

# C Code

```c
#include <stdio.h>

/*
 * Compute AP and PB (vertical distances) and x-coordinate of P (
     using section formula).
 *
 * Inputs:
 * Ax, Ay, Bx, By, yP
 *
 * Outputs (via pointers):
 * *ratio_AP : AP (vertical distance Ay - yP)
 * *ratio_PB : PB (vertical distance yP - By)
 * *xP : x-coordinate of P computed by section formula
 */
void section_point(double Ax, double Ay, double Bx, double By,
    double yP,
                   double *ratio_AP, double *ratio_PB, double *xP)
                       {
```

# C Code

```c
double m = Ay - yP; /* vertical distance AP */
double n = yP - By; /* vertical distance PB */

if (m + n == 0.0) {
    /* degenerate: cannot determine location */
    fprintf(stderr, Error: m + n == 0, cannot compute section
        .\n);
    if (ratio_AP) *ratio_AP = 0.0;
    if (ratio_PB) *ratio_PB = 0.0;
    if (xP) *xP = 0.0;
    return;
}

if (ratio_AP) *ratio_AP = m;
if (ratio_PB) *ratio_PB = n;

/* Section formula for internal division:
   x = (n*Ax + m*Bx) / (m + n)
```

# C Code

```
        (because AP:PB = m:n, weight on A is n, on B is m)
    */
    if (xP) *xP = (n*Ax + m*Bx) / (m + n);
}
```

# Python Code through shared output

```
#!/usr/bin/env python3
import os
import subprocess
import ctypes
import math
import matplotlib.pyplot as plt

C_FILE = section.c
SO_FILE = ./libsection.so

# Auto-compile if shared lib not present
if not os.path.exists(SO_FILE):
    print(libsection.so not found  compiling section.c ...)
    cmd = [gcc, -shared, -o, libsection.so, -fPIC, C_FILE]
    try:
        subprocess.run(cmd, check=True)
        print(Compiled libsection.so)
```

# Python Code through shared output

```
    except subprocess.CalledProcessError as e:
        print(Compilation failed:, e)
        raise SystemExit(1)

# Load shared library
lib = ctypes.CDLL(SO_FILE)

# Set function signature:
# void section_point(double, double, double, double, double,
# double*, double*, double*)
lib.section_point.argtypes = [ctypes.c_double, ctypes.c_double,
                              ctypes.c_double, ctypes.c_double,
                              ctypes.c_double,
                              ctypes.POINTER(ctypes.c_double),
                              ctypes.POINTER(ctypes.c_double),
                              ctypes.POINTER(ctypes.c_double)]
lib.section_point.restype = None
```

# Python Code through shared output

```
# Input points
Ax, Ay = -5.0, 8.0
Bx, By = 4.0, -10.0
yP = 4.0

# Prepare output holders
ratio_AP = ctypes.c_double()
ratio_PB = ctypes.c_double()
xP = ctypes.c_double()

# Call C function
lib.section_point(Ax, Ay, Bx, By, yP,
                  ctypes.byref(ratio_AP),
                  ctypes.byref(ratio_PB),
                  ctypes.byref(xP))

# Read outputs
m = ratio_AP.value # AP vertical distance (Ay - yP)
n = ratio_PB.value # PB vertical distance (yP - By)
```

# Python Code through shared output

```
x_val = xP.value

# Convert ratio to smallest integer ratio (if sensible)
# We'll round to nearest integer then reduce by gcd if both
    nonzero
def reduced_ratio(a, b):
    ia = int(round(a))
    ib = int(round(b))
    if ia == 0 and ib == 0:
        return (0, 0)
    if ia < 0: ia = -ia
    if ib < 0: ib = -ib
    g = math.gcd(ia, ib) if (ia != 0 and ib != 0) else (ia or ib)
    if g == 0:
        return (ia, ib)
    return (ia // g, ib // g)

ratA_int, ratB_int = reduced_ratio(m, n)
```

# Python Code through shared output

```
print(f Raw m (AP) = {m}, n (PB) = {n})
print(f AP:PB (reduced) = {ratA_int}:{ratB_int})
print(f x = {x_val})

# ---- Plot ----
A = (Ax, Ay)
B = (Bx, By)
P = (x_val, yP)

plt.plot([A[0], B[0]], [A[1], B[1]], linestyle='--', label=Line
    AB)
plt.scatter(*A, marker='o', label=fA{A}, zorder=5)
plt.scatter(*B, marker='o', label=fB{B}, zorder=5)
plt.scatter(*P, marker='o', label=fP({x_val:.3g},{yP}), zorder=5)

plt.text(A[0]-0.6, A[1]+0.4, fA{A}, color=red)
plt.text(B[0]+0.4, B[1]-0.6, fB{B}, color=blue)
```

# Python Code through shared output

```
plt.text(P[0]+0.4, P[1]+0.4, fP({x_val:.3g},{yP}), color=green)

plt.axhline(0, color=gray, lw=0.5)
plt.axvline(0, color=gray, lw=0.5)
plt.grid(True, linestyle=--, alpha=0.5)
plt.xlabel(x)
plt.ylabel(y)
plt.title(fP divides AB in ratio {ratA_int}:{ratB_int})
plt.legend()
plt.gca().set_aspect('equal', adjustable='box')
plt.show()
```

# Plot by python using shared output from c