1.5.28

Naman kumar-EE25BTECH11041

August 26,2025

Question

P$(5, -3)$ and **Q** $(3,\ y)$ are the points of trisection of the line segment joining **A**$(7, -2)$ and **B**$(1, -5)$. Then y equals.

Equation Used

$$\mathbf{Q} = \frac{1}{1+k}\left(\mathbf{A} + k\mathbf{B}\right) \tag{1}$$

Theoratical Solution

$$\mathbf{Q} = \frac{1}{1+2}\left(\begin{pmatrix}7\\-2\end{pmatrix} + 2\begin{pmatrix}1\\-5\end{pmatrix}\right) \tag{2}$$

$$\mathbf{Q} = \frac{1}{1+2}\left(\begin{pmatrix}9\\-12\end{pmatrix}\right) \tag{3}$$

$$\mathbf{Q} = \begin{pmatrix}3\\-4\end{pmatrix} \tag{4}$$

$$\mathbf{Q} = \begin{pmatrix}3\\y\end{pmatrix} = \begin{pmatrix}3\\-4\end{pmatrix} \tag{5}$$

C Code - Section formula function

```c
#include <stdio.h>

void trisec(double x1, double y1, double x2, double y2, double* a
    , double* b, double* c, double* d){
    *a = (x1+2*x2)/3;
    *b = (y1+2*y2)/3;
    *c = (2*x1+x2)/3;
    *d = (2*y1+y2)/3;
}
```

Python Code through shared output

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

# --- Ctypes Setup ---

# Load the shared library.
# Make sure 'main.so' is in the same directory as this Python
    script,
# or provide the full path to it.
try:
    c_lib = ctypes.CDLL('./main.so')
except OSError as e:
    print(f"Error loading shared library: {e}")
    print("Please ensure 'main.so' is in the same directory as
        this script.")
    exit()

# Define the argument types for the C function.
```

Python Code through shared output

```
# The C function signature is:
# void trisec(double x1, double y1, double x2, double y2, double*
    a, double* b, double* c, double* d)
c_lib.trisec.argtypes = [
    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),
    ctypes.POINTER(ctypes.c_double)
]

# Define the return type of the function.
c_lib.trisec.restype = None

# --- Calculation ---
```

Python Code through shared output

```
# Define the input coordinates for the two endpoints of the line
    segment
x1, y1 = 7.0, -2.0
x2, y2 = 1.0, -5.0

# Prepare ctypes variables to hold the results.
# These will act as the pointers that the C function will write
    to.
ta = ctypes.c_double()
tb = ctypes.c_double()
tc = ctypes.c_double()
td = ctypes.c_double()

# Call the C function from Python to calculate the trisection
    point
c_lib.trisec(x1, y1, x2, y2, ctypes.byref(ta), ctypes.byref(tb),
    ctypes.byref(tc), ctypes.byref(td))
```

Python Code through shared output

```
# Extract the float values from the ctypes variables
ta_val, tb_val , tc_val, td_val= ta.value, tb.value, tc.value, td
    .value
print(f"Line segment from ({x1}, {y1}) to ({x2}, {y2})")
print(f"Trisection point 1 calculated by C code: ({ta_val:.2f}, {
    tb_val:.2f})")
print(f"Trisection point 2 calculated by C code: ({tc_val:.2f}, {
    td_val:.2f})")


# --- Plotting ---

# Create the plot
plt.figure(figsize=(8, 6))

# Plot the full line segment
plt.plot([x1, x2], [y1, y2], 'g--', label="Line Segment")
```

Python Code through shared output

```
# Plot the endpoints of the line
plt.scatter([x1, x2], [y1, y2], color="red", s=100, zorder=5,
    label="Endpoints")
plt.text(x1, y1 - 0.5, f"A ({x1:.1f}, {y1:.1f})", color="red",
    fontsize=10)
plt.text(x2, y2 - 0.5, f"B ({x2:.1f}, {y2:.1f})", color="red",
    fontsize=10)

# Plot the calculated trisection point
plt.scatter(ta_val, tb_val, color="blue", marker="X", s=150,
    zorder=5, label="Trisection Point")
plt.text(ta_val, tb_val + 0.3, f"Trisection Pt 1\n({ta_val:.2f},
    {tb_val:.2f})", color="blue", fontsize=10)
plt.scatter(tc_val, td_val, color="blue", marker="X", s=150,
    zorder=5, label="Trisection Point")
plt.text(tc_val, td_val + 0.3, f"Trisection Pt 2\n({tc_val:.2f},
    {td_val:.2f})", color="blue", fontsize=10)
```

Python Code through shared output

```
# Configure plot appearance
plt.title("Line Segment and its Trisection Point")
plt.xlabel("X-axis")
plt.ylabel("Y-axis")
plt.legend(loc="upper left")
plt.grid(True)
plt.axis("equal") # Ensures the scaling is the same on both axes
plt.show()
```

Python Code: Direct

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

A = (7, -2)
B = (1, -5)
P = (5, -3)
Q = (3, -4)

fig, ax = plt.subplots(figsize=(10, 7))

ax.plot([A[0], B[0]], [A[1], B[1]], color='skyblue', linestyle='-
    ', linewidth=2, label='Line Segment AB')

points = {'A': A, 'B': B, 'P': P, 'Q': Q}
colors = {'A': 'blue', 'B': 'blue', 'P': 'red', 'Q': 'green'}

for name, (x, y) in points.items():
    ax.scatter(x, y, s=100, color=colors[name], zorder=5)
    ax.text(x + 0.15, y + 0.15, f'{name}({x}, {y})', fontsize=12,
```

Python Code: Direct

```
ax.set_title('Trisection of a Line Segment', fontsize=16,
    fontweight='bold')
ax.set_xlabel('X-axis', fontsize=14)
ax.set_ylabel('Y-axis', fontsize=14)

ax.grid(True, linestyle='--', alpha=0.6)

ax.set_xlim(0, 8)
ax.set_ylim(-6, 0)

ax.set_aspect('equal', adjustable='box')

ax.legend()

plt.savefig('trisection_plot.png')
plt.show()
```