Table 5.7.7

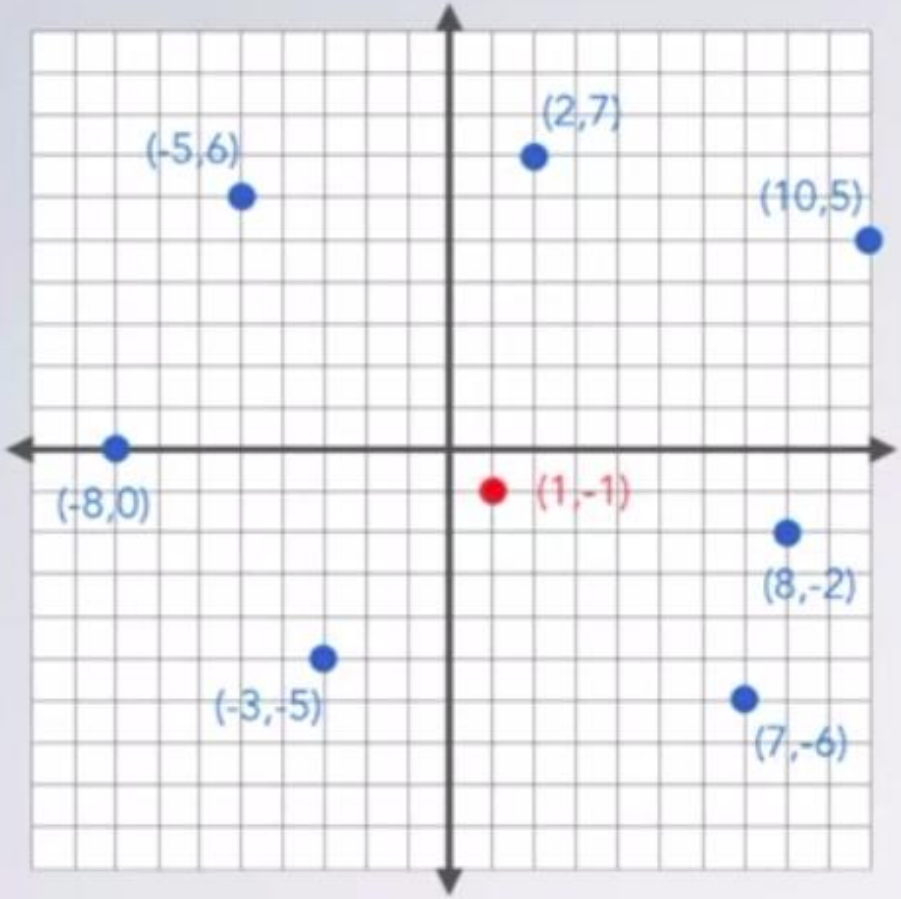
Program 5.7.1 : Write an assembly language program to add two 8 bit numbers in TMS320C5X.

Program :

```
.MMREGS
.TEXT
START:
LDP #100H       ; Initialize data pointer to the location 8000H at page 100H
LACC 0H         ; Load the accumulator with first data from 8000H
ADD #3H         ; Add the second data with accumulator content.
SACL 2H         ; Store the result in 8002H
H:  B  H        ; End program with loop
```

Program 5.7.2 : Write an assembly language program to 2's complement of a given number in TMS320C5X

Step 1: Do an instance of the problem

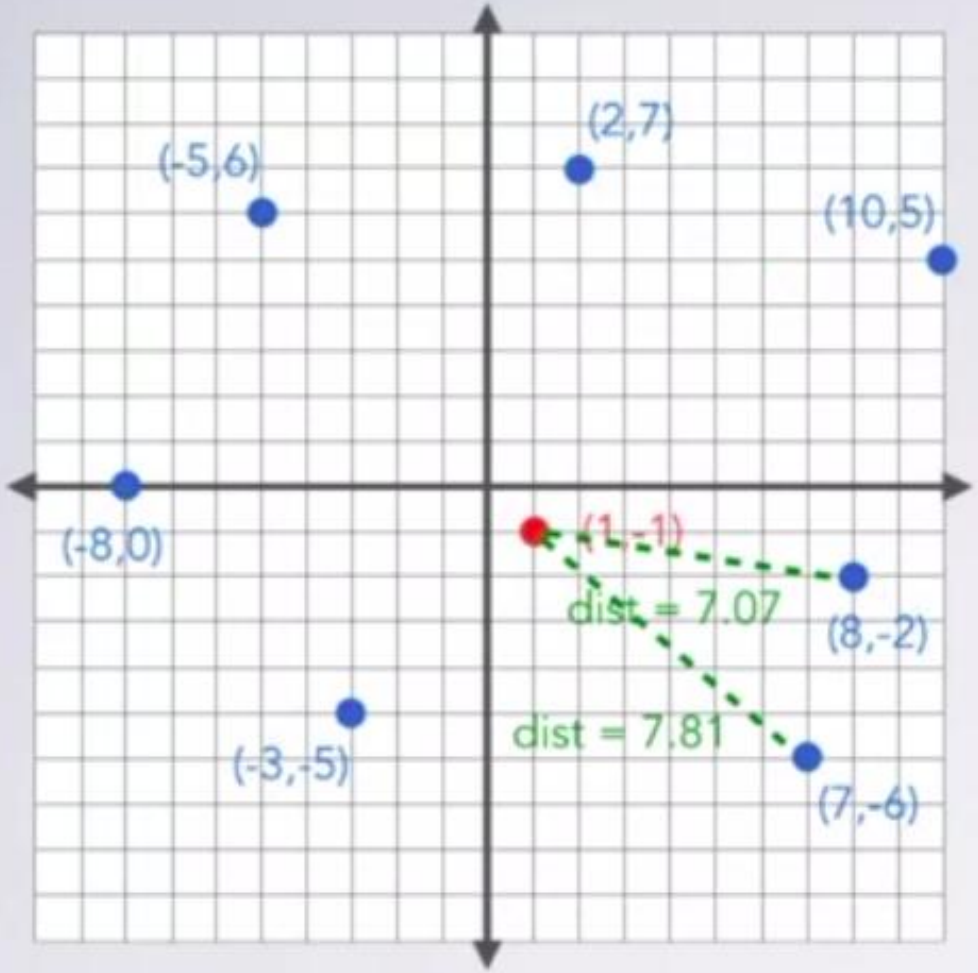


I pick:

S = { (2,7), (10,5), (8,-2), (7,-6),
(-3,-5), (-8,0), (-5,6) }

P = (1,-1)

Step 1: Do an instance of the problem



I pick:

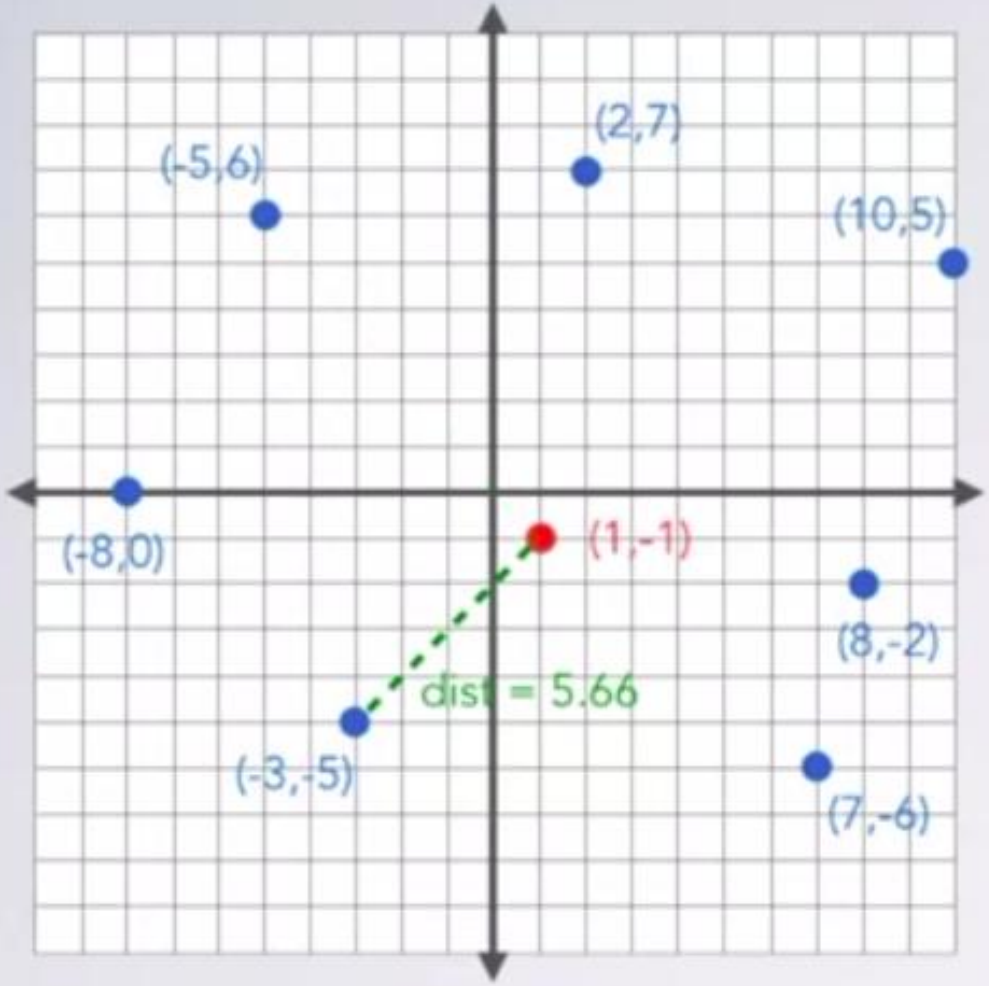
S = { (2,7), (10,5), (8,-2), (7,-6), (-3,-5), (-8,0), (-5,6) }

P = (1,-1)

Domain knowledge:

distance = $\sqrt{\Delta x^2 + \Delta y^2}$

Step 1: Do an instance of the problem



I pick:

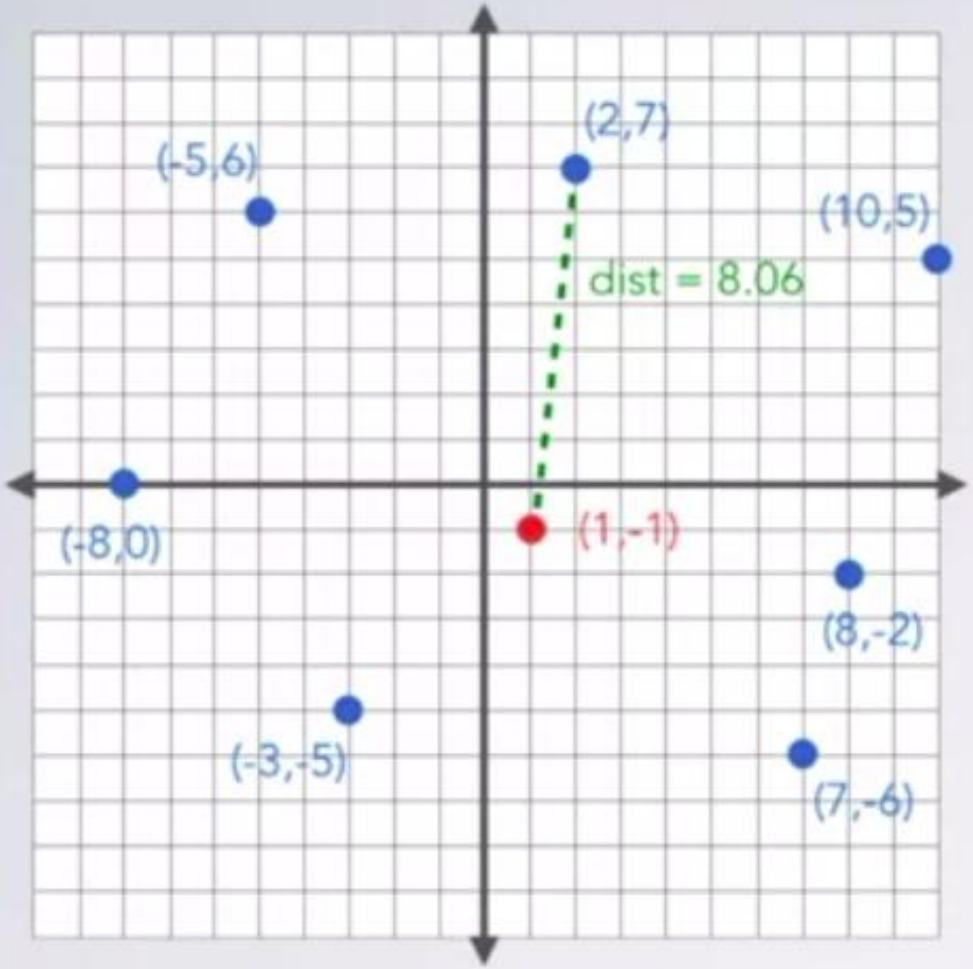
$S = \{ (2,7), (10,5), (8,-2), (7,-6), (-3,-5), (-8,0), (-5,6) \}$

$P = (1,-1)$

Domain knowledge:

distance $= \sqrt{\Delta x^2 + \Delta y^2}$

Step 2: Write down what you just did

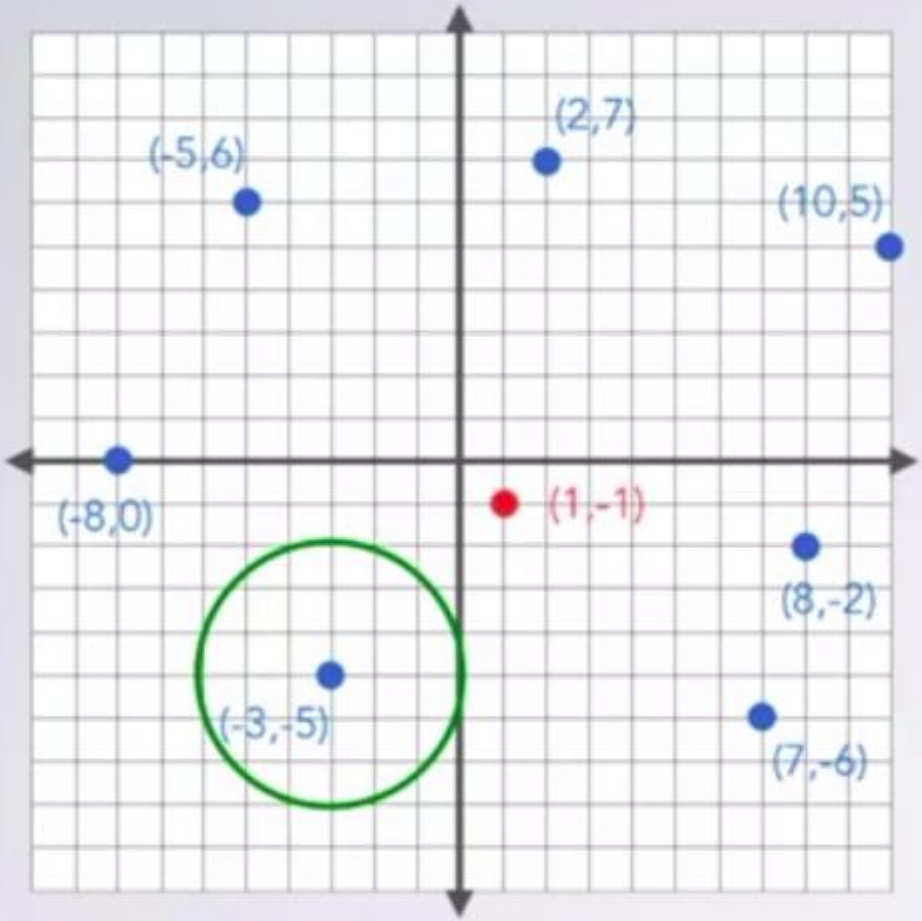


Computed $\sqrt{1^2 + 8^2} = 8.06$

Computed $\sqrt{9^2 + 6^2} = 10.82$

Compared 10.82 to 8.06—8.06 is smaller

Step 2: Write down what you just did



Computed $\sqrt{1^2 + 8^2} = 8.06$

Computed $\sqrt{9^2 + 6^2} = 10.82$

Compared 10.82 to 8.06—8.06 is smaller

Computed $\sqrt{7^2 + -1^2} = 7.07$

Compared 7.07 to 8.06—7.07 is smaller

Computed $\sqrt{6^2 + -5^2} = 7.81$

Compared 7.81 to 7.07—7.07 is smaller

Computed $\sqrt{-4^2 + -4^2} = 5.66$

Compared 5.66 to 7.07—5.66 is smaller

Computed $\sqrt{-9^2 + 1^2} = 9.06$

Compared 9.06 to 5.66—5.66 is smaller

Computed $\sqrt{-6^2 + 7^2} = 9.22$

Compared 9.22 to 5.66—5.66 is smaller

I gave an answer of (-3,-5)

Step 2: Write down what you just did

These seem like reasonable steps, but…

We glossed over something we did.

Computed $\sqrt{1^2 + 8^2} = 8.06$

Computed $\sqrt{9^2 + 6^2} = 10.82$

Compared 10.82 to 8.06—8.06 is smaller

Computed $\sqrt{7^2 + -1^2} = 7.07$

Compared 7.07 to 8.06—7.07 is smaller

Computed $\sqrt{6^2 + -5^2} = 7.81$

Compared 7.81 to 7.07—7.07 is smaller

Computed $\sqrt{-4^2 + -4^2} = 5.66$

Compared 5.66 to 7.07—5.66 is smaller

Computed $\sqrt{-9^2 + 1^2} = 9.06$

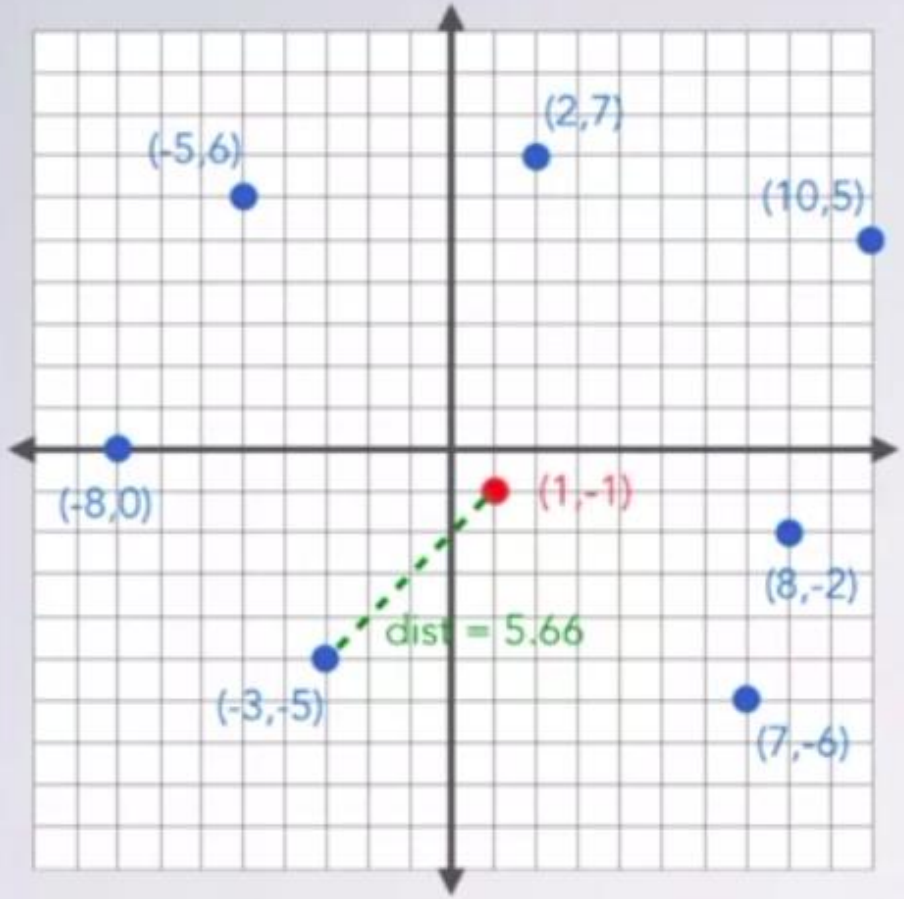
Compared 9.06 to 5.66—5.66 is smaller

Computed $\sqrt{-6^2 + 7^2} = 9.22$

Compared 9.22 to 5.66—5.66 is smaller

Here is an important clue → I gave an answer of (-3,-5)

Step 2: Write down what you just did



Computed $\sqrt{1^2 + 8^2} = 8.06$

Started with best choice of (2,7)

Computed $\sqrt{9^2 + 6^2} = 10.82$

Compared 10.82 to 8.06—8.06 is smaller

Computed $\sqrt{7^2 + -1^2} = 7.07$

Compared 7.07 to 8.06—7.07 is smaller

Updated best choice to (8,-2)

Computed $\sqrt{6^2 + -5^2} = 7.81$

Compared 7.81 to 7.07—7.07 is smaller

Computed $\sqrt{-4^2 + -4^2} = 5.66$

Compared 5.66 to 7.07—5.66 is smaller

Updated best choice to (-3,-5)

Computed $\sqrt{-9^2 + 1^2} = 9.06$

Compared 9.06 to 5.66—5.66 is smaller

Computed $\sqrt{-6^2 + 7^2} = 9.22$

Compared 9.22 to 5.66—5.66 is smaller

I gave an answer of (-3,-5)

Now ready to generalize!

Step 3: Generalize these steps

Computed $\sqrt{1^2 + 8^2} = 8.06$

Started with best choice of (2,7)

Computed $\sqrt{9^2 + 6^2} = 10.82$

Compared 10.82 to 8.06—8.06 is smaller

Computed $\sqrt{7^2 + -1^2} = 7.07$

Compared 7.07 to 8.06—7.07 is smaller

Updated best choice to (8,-2)

Computed $\sqrt{6^2 + -5^2} = 7.81$

Compared 7.81 to 7.07—7.07 is smaller

Computed $\sqrt{-4^2 + -4^2} = 5.66$

Compared 5.66 to 7.07—5.66 is smaller

Updated best choice to (-3,-5)

Computed $\sqrt{-9^2 + 1^2} = 9.06$

Compared 9.06 to 5.66—5.66 is smaller

Computed $\sqrt{-6^2 + 7^2} = 9.22$

Compared 9.22 to 5.66—5.66 is smaller

I gave an answer of (-3,-5)

We have some similar steps,
but we need to make them match up

Step 3: Generalize these steps

Computed $\sqrt{1^2 + 8^2} = 8.06$

Started with best choice of (2,7)

Computed $\sqrt{(S_1\text{'s } x - P\text{'s } x)^2 + (S_1\text{'s } y - P\text{'s } y)^2} = \mathbf{10.82}$

Compared 10.82 to 8.06—8.06 is smaller

Computed $\sqrt{(S_2\text{'s } x - P\text{'s } x)^2 + (S_2\text{'s } y - P\text{'s } y)^2} = \mathbf{7.07}$

Compared 7.07 to 8.06—7.07 is smaller

Updated best choice to (8,-2)

Computed $\sqrt{(S_3\text{'s } x - P\text{'s } x)^2 + (S_3\text{'s } y - P\text{'s } y)^2} = \mathbf{7.81}$

Compared 7.81 to 7.07—7.07 is smaller

Computed $\sqrt{(S_4\text{'s } x - P\text{'s } x)^2 + (S_4\text{'s } y - P\text{'s } y)^2} = \mathbf{5.66}$

Compared 5.66 to 7.07—5.66 is smaller

Updated best choice to (-3,-5)

Computed $\sqrt{(S_5\text{'s } x - P\text{'s } x)^2 + (S_5\text{'s } y - P\text{'s } y)^2} = \mathbf{9.06}$

Compared 9.06 to 5.66—5.66 is smaller

Computed $\sqrt{(S_6\text{'s } x - P\text{'s } x)^2 + (S_6\text{'s } y - P\text{'s } y)^2} = \mathbf{9.22}$

Compared 9.22 to 5.66—5.66 is smaller

I gave an answer of (-3,-5)

Step 3: Generalize these steps

Computed $\sqrt{(S_0\text{'s } x - P\text{'s } x)^2 + (S_0\text{'s } y - P\text{'s } y)^2}$ call it **bestDistance**

Started with best choice of $\mathbf{S_0}$

Computed $\sqrt{(S_1\text{'s } x - P\text{'s } x)^2 + (S_1\text{'s } y - P\text{'s } y)^2}$ call it **currentDistance**

Compared **currentDistance** to **bestDistance**—**bestDistance** is smaller

Computed $\sqrt{(S_2\text{'s } x - P\text{'s } x)^2 + (S_2\text{'s } y - P\text{'s } y)^2}$ call it **currentDistance**

Compared **currentDistance** to **bestDistance**—**currentDistance** is smaller

Updated best choice to $\mathbf{S_2}$ and **bestDistance** to **currentDistance**

Computed $\sqrt{(S_3\text{'s } x - P\text{'s } x)^2 + (S_3\text{'s } y - P\text{'s } y)^2}$ call it **currentDistance**

Compared **currentDistance** to **bestDistance**—**bestDistance** is smaller

Computed $\sqrt{(S_4\text{'s } x - P\text{'s } x)^2 + (S_4\text{'s } y - P\text{'s } y)^2}$ call it **currentDistance**

Compared **currentDistance** to **bestDistance**—**currentDistance** is smaller

Updated best choice to $\mathbf{S_4}$ and **bestDistance** to **currentDistance**

Computed $\sqrt{(S_5\text{'s } x - P\text{'s } x)^2 + (S_5\text{'s } y - P\text{'s } y)^2}$ call it **currentDistance**

Compared **currentDistance** to **bestDistance**—**bestDistance** is smaller

Computed $\sqrt{(S_6\text{'s } x - P\text{'s } x)^2 + (S_6\text{'s } y - P\text{'s } y)^2}$ call it **currentDistance**

Compared **currentDistance** to **bestDistance**—**bestDistance** is smaller

I gave an answer of (-3,-5)

Computed $\sqrt{(S_0\text{'s } x - P\text{'s } x)^2 + (S_0\text{'s } y - P\text{'s } y)^2}$ call it **bestDistance**

Started with best choice of S_0

Computed $\sqrt{(S_1\text{'s } x - P\text{'s } x)^2 + (S_1\text{'s } y - P\text{'s } y)^2}$ call it **currentDistance**

If **currentDistance** is smaller than **bestDistance**

Then update **bestChoice** to S_1 and **bestDistance** to **currentDistance**

Computed $\sqrt{(S_2\text{'s } x - P\text{'s } x)^2 + (S_2\text{'s } y - P\text{'s } y)^2}$ call it **currentDistance**

If **currentDistance** is smaller than **bestDistance**

Then update **bestChoice** to S_2 and **bestDistance** to **currentDistance**

Computed $\sqrt{(S_3\text{'s } x - P\text{'s } x)^2 + (S_3\text{'s } y - P\text{'s } y)^2}$ call it **currentDistance**

If **currentDistance** is smaller than **bestDistance**

Then update **bestChoice** to S_3 and **bestDistance** to **currentDistance**

Computed $\sqrt{(S_4\text{'s } x - P\text{'s } x)^2 + (S_4\text{'s } y - P\text{'s } y)^2}$ call it **currentDistance**

If **currentDistance** is smaller than **bestDistance**

Then update **bestChoice** to S_4 and **bestDistance** to **currentDistance**

Computed $\sqrt{(S_5\text{'s } x - P\text{'s } x)^2 + (S_5\text{'s } y - P\text{'s } y)^2}$ call it **currentDistance**

If **currentDistance** is smaller than **bestDistance**

Then update **bestChoice** to S_5 and **bestDistance** to **currentDistance**

Computed $\sqrt{(S_6\text{'s } x - P\text{'s } x)^2 + (S_6\text{'s } y - P\text{'s } y)^2}$ call it **currentDistance**

If **currentDistance** is smaller than **bestDistance**

Then update **bestChoice** to S_6 and **bestDistance** to **currentDistance**

I gave an answer of (-3,-5)

Step 3: Generalize these steps

Compute $\sqrt{(S_0\text{'s } x - P\text{'s } x)^2 + (S_0\text{'s } y - P\text{'s } y)^2}$ call it **bestDistance**

Start with best choice of $\mathbf{S_0}$

Count from 1 to the number of points in S exclusive, call each number **i**

Compute $\sqrt{(S_i\text{'s } x - P\text{'s } x)^2 + (S_i\text{'s } y - P\text{'s } y)^2}$ call it **currentDistance**

If **currentDistance** is smaller than **bestDistance**

Then update **bestChoice** to $\mathbf{S_i}$ and **bestDistance** to **currentDistance**

I gave an answer of (-3,-5)

Step 3: Generalize these steps

Compute $\sqrt{(S_0\text{'s } x - P\text{'s } x)^2 + (S_0\text{'s } y - P\text{'s } y)^2}$ call it **bestDistance**

Start with best choice of S_0

Count from 1 to the number of points in S exclusive, call each number **i**

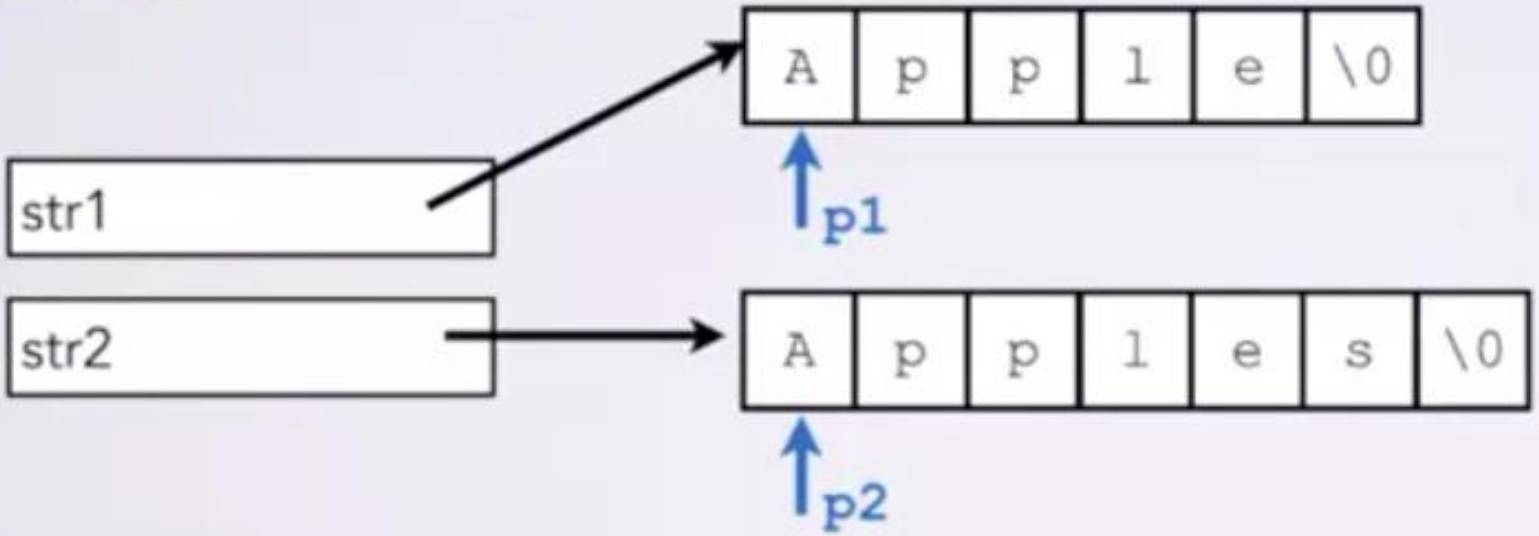
- Compute $\sqrt{(S_i\text{'s } x - P\text{'s } x)^2 + (S_i\text{'s } y - P\text{'s } y)^2}$ call it **currentDistance**
- If **currentDistance** is smaller than **bestDistance**
  - Then update **bestChoice** to S_i and **bestDistance** to **currentDistance**

Give an answer of **bestChoice**

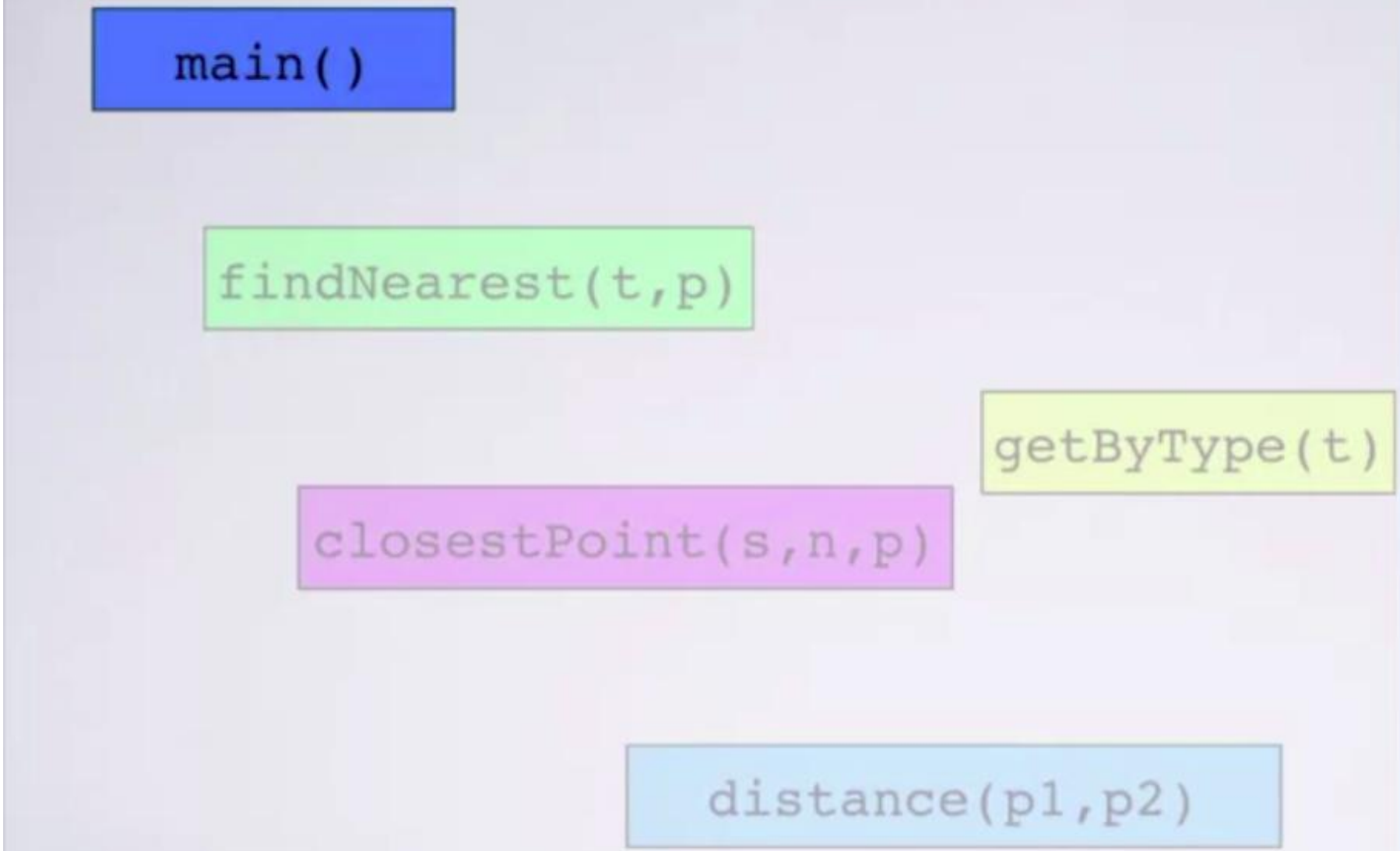
Corner case when S has 0 points?

Code Needs to Match Algorithm

I made an arrow (p1) pointing at the first letter of `str1`
I made an arrow (p2) pointing at the first letter of `str2`
As long as what p1 points at is the same as what p2 points at
    If p1 points at '\0'
        Then answer "yes"
    Advance `p1` to point at the next letter
    Advance `p2` to point at the next letter
Answer "no"

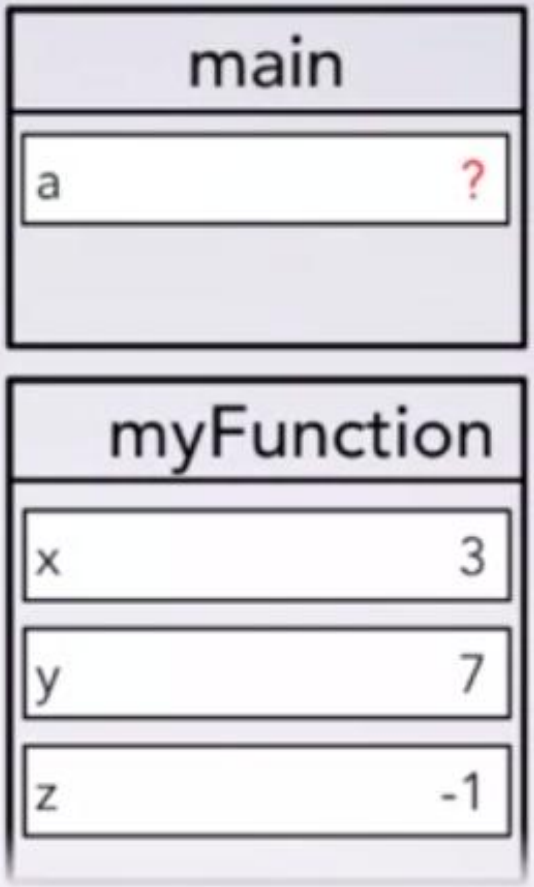


Combining Many Functions



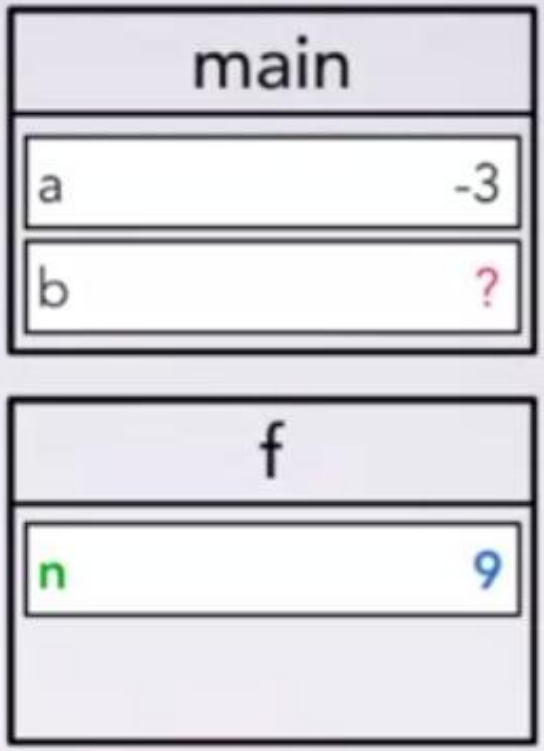
```
int myFunction (int x, int y) {
  int z  = 2 * x - y;
  return z * x;
}
int f (int n) {
  return 3 + myFunction(n, n+1);
}
int main (void) {
  int a;
  a = myFunction(3, 7);
  int b = f(a * a);
  return 0;
}
```

-3

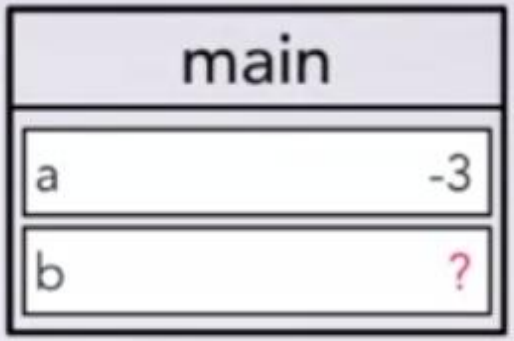


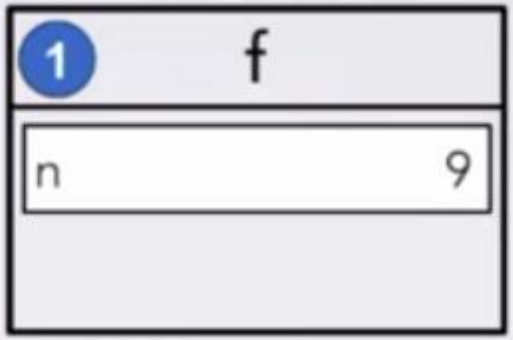
1. Compute *return value*
2. Find where to return (noted in frame)
3. Copy return value to call site
4. Move execution arrow back to call site
5. Destroy frame

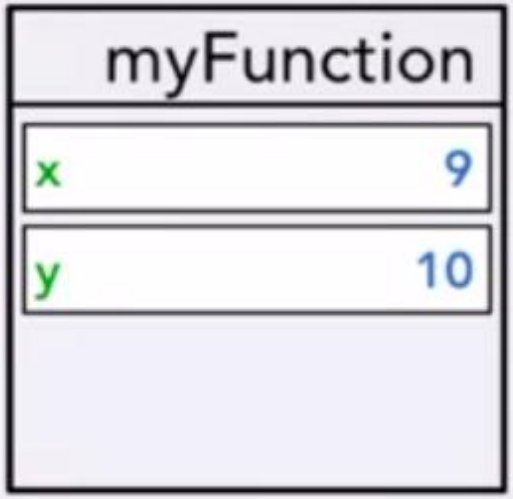
```
int myFunction (int x, int y) {
  int z  = 2 * x - y;
  return z * x;
}
int f (int n) {
  return 3 + myFunction(n, n+1);
}
int main (void) {
  int a;
  a = myFunction(3, 7);
  int b = f(a * a);
  return 0;
}
```



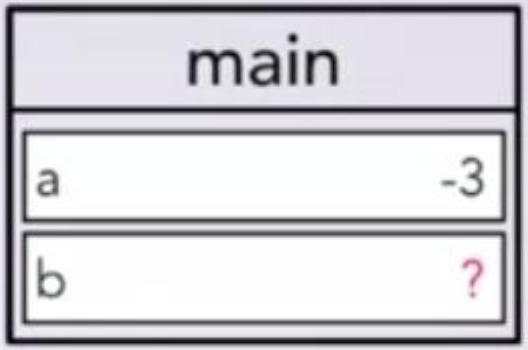
```
int myFunction (int x, int y) {
  int z  = 2 * x - y;
  return z * x;
}
int f (int n) {
  return 3 + myFunction(n, n+1);
}
int main (void) {
  int a;
  a = myFunction(3, 7);
  int b = (a * a);
  return 0;
}
```





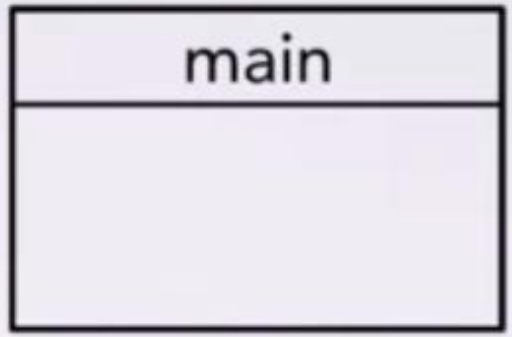


```
int myFunction (int x, int y) {
  int z  = 2 * x - y;
  return z * x;
}
int f (int n) {
  return 3 + myFunction(n, n+1);
}
int main (void) {
  int a;
  a = myFunction(3, 7);
  int b = f(a * a);   75
  return 0;
}
```



Function `printf` prints output to the terminal

```
int main(void) {
  int a = 42;
  printf("Hello World\n");
  printf("a\n");
  return 0;
}
```



Output

I

Use format specifiers with `printf` for formatted output

- `%d` formats integers as decimal numbers

```
int main(void) {
  int a = 42;
  int b = 7;
  printf("a is %d\n", a);
  printf("b is %d, and a + b is %d\n", b, a+b);
  return 0;
}
```

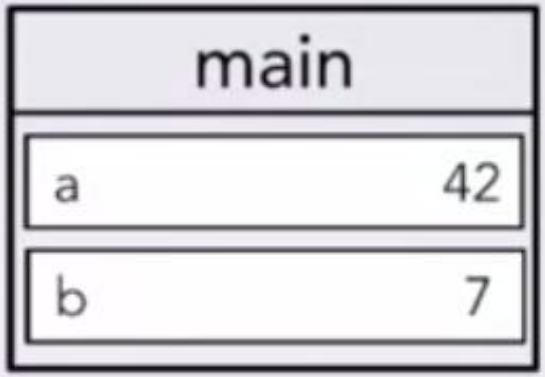
Output

```
a is 42
b is 7, and a + b is 49
```

Use format specifiers with `printf` for formatted output

- `%d` formats integers as decimal numbers

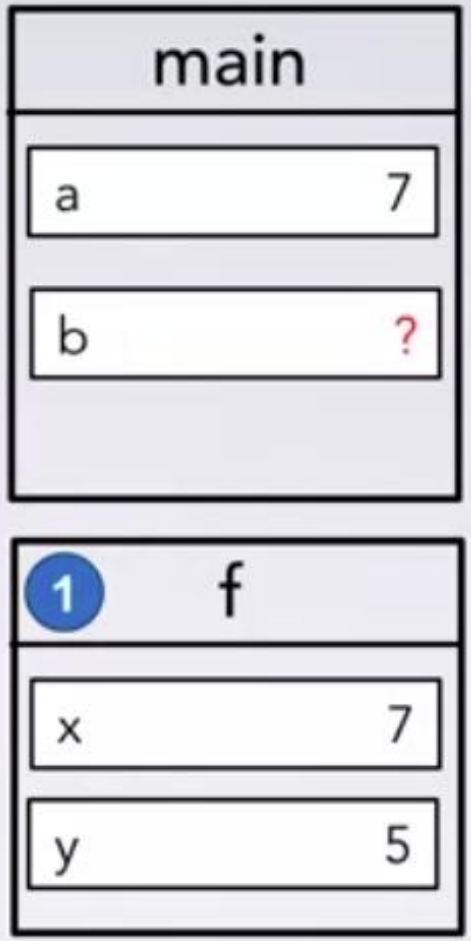
```
int main(void) {
  int a = 42;
  int b = 7;
  printf("a is %d\n", a);
  printf("b is %d, and a + b is %d\n", b, a+b);
  return 0;
}
```



Output

```
a is 42
b is 7, and a + b is 49
```

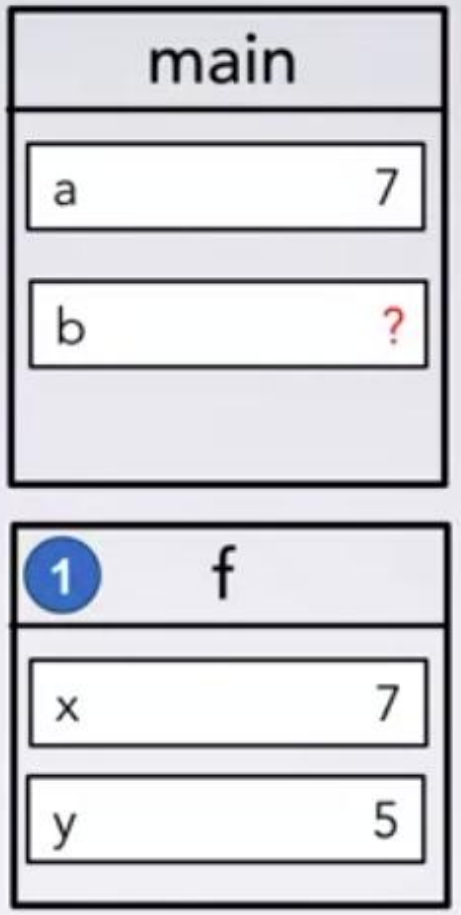
```
int f (int x, int y) {
  if (x < y) {
     printf("x < y\n");
     return y + x;
  }
  else {
     printf("x >= y\n");
     if (x > 8) {
        return y + 7;
     }
  }
  return x - 2;
}

int main (void) {
  int a = f (3, 4);
  int b = f (a, 5);
  return 0;
}
```



Output

x < y

```
int f (int x, int y) {
  if (x < y) {
     printf("x < y\n");
     return y + x;
  }
  else {
      printf("x >= y\n");
      if (x > 8) {
         return y + 7;
      }
  }
  return x - 2;
}

int main (void) {
  int a = f (3, 4);
  int b = 1 (a, 5);
  return 0;
}
```

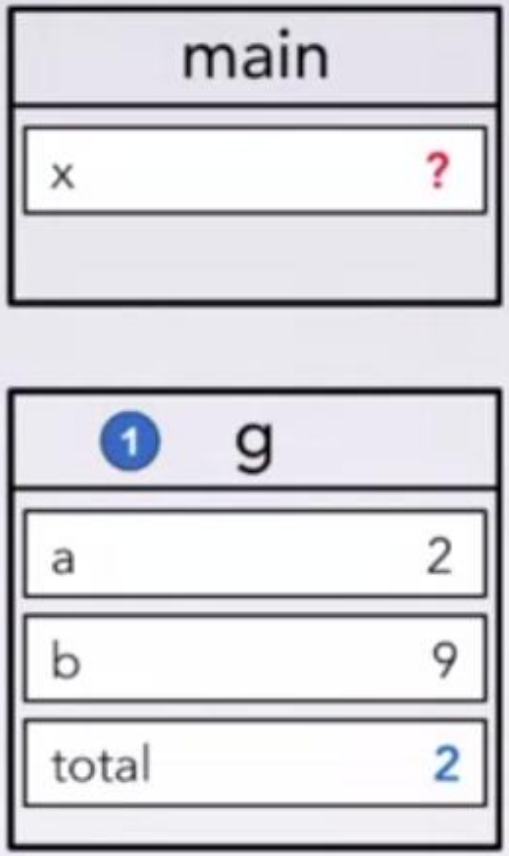


Output

```
x < y
x >= y
```

```
int g (int n, int x) {
  switch (x + n) {
    case 8:
      x = x + 1;
    case 0:
      n = n - 1;
      break;
    case 14:
      return n - x;
    default:
      x = n;
      break;
  }
  return x * n;
}

int main (void) {
  int a = g(10, 4);
  int b = g(a, 2);
  int c = g(9, b);
  return 0;
}
```

```
int g(int a, int b) {
  int total = 0;
  while (a < b) {
    total += a;
    printf("a=%d, b=%d\n", a, b);
    a++;
    b--;
  }
  return total;
}

int main (void) {
  int x = g(2, 9);
  printf("x=%d\n", x);
  return 0;
}
```



Output

Equivalent loops

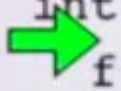
For loop

```
int main(void) {
  int y = 1;
  int n = 3;
  for (int i = 1; i < n; i++) {
    y = y * i;
  }
  return 0;
}
```

While loop

```
int main(void) {
  int y = 1;
  int n = 3;
  {
    int i = 1;
    while (i < n) {
      y = y * i;
      i++;
    }
  }
  return 0;
}
```

```
void f (int a, int b) {
  while (a < b) {
    printf ("a=%d, b=%d\n", a, b);
    if (a % 2 == 0) {
      for (int i = a; i < b; i++) {
        printf("i=%d\n", i);
      }
    }
    a++;
    b--;
  }
}

int main (void) {
  f(3, 8);
  return 0;
}
```



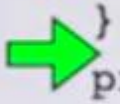
main

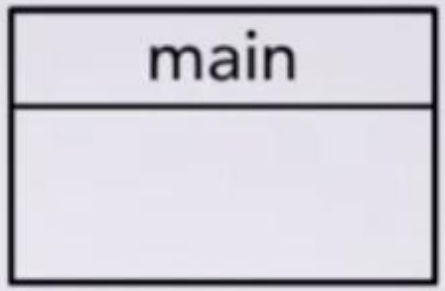
Output

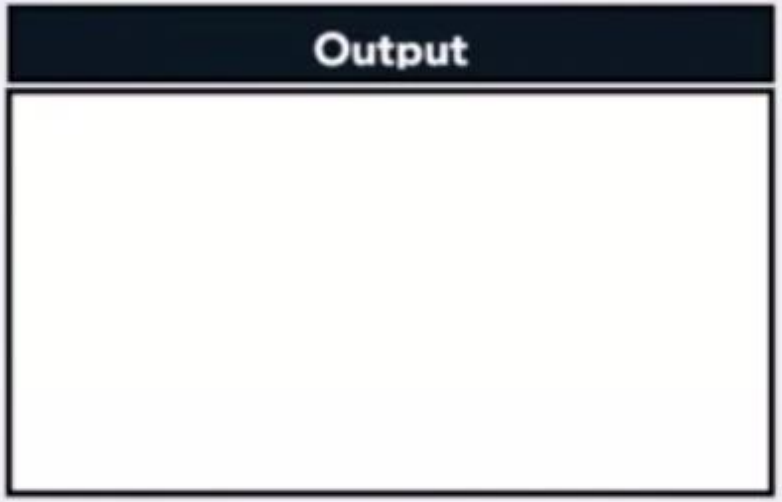
```
void printRemainders (int lo, int hi, int n) {
  {
  int i = lo;
    for ( ; i < hi; i++) {
      if (i == 0) {
        printf("Cannot divide by 0.\n");
        continue;
      }
      printf("%d mod %d = %d\n", n, i, n % i);
    }
  }
}

int main (void) {
  printRemainders(-2, 4, 5);
  return 0;
}
```

Transform for to while loop

```
void printRemainders (int lo, int hi, int n) {
  int i = lo;
  while (i < hi) {
    if (i == 0) {
      printf("Cannot divide by 0.\n");
      i++;
      continue;
    }
    printf("%d mod %d = %d\n", n, i, n % i);
    i++;
  }
}

int main (void) {
  printRemainders(-2, 4, 5);
  return 0;
}
```







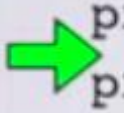
```
char letter = 'G';
int negNumber = -1;
unsigned int age = 33;

printf("My name begins with %c\n", letter);
printf("Look, I'm negative! --> %d\n", negNumber);
printf("I am %d years old!\n", age);
printf("\t in octal (base 8) = %o\n", age);
printf("\t in hex (base 16) = %x\n", age);
```

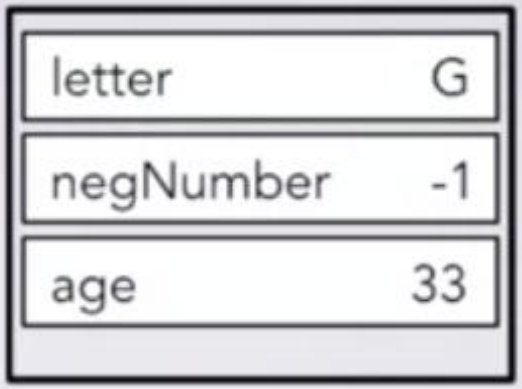
letter	G
negNumber	-1
age	33

Output

```
My name begins with G
Look, I'm negative! --> –1
I am 33 years old!
        in octal (base 8) = 41
        in hex (base 16) = 21
```

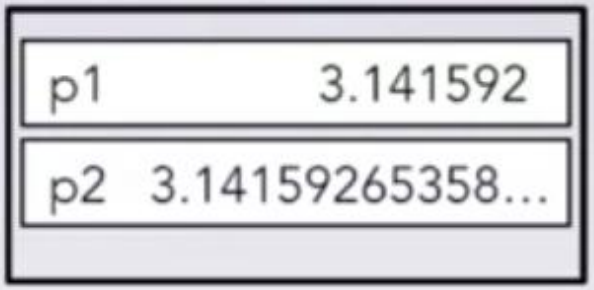
```
char letter = 'G';
int negNumber = -1;
unsigned int age = 33;

printf("G's numeric value is %d\n", letter);
printf("-1 as hex is %x\n", negNumber);
printf("-1 as an unsigned is %u\n", negNumber);
printf("33 as a letter is %c\n", age);
```





Output

```
G's numeric value is 71
-1 as hex is ffffffff
-1 as an unsigned is 4294967295
```




```
float p1 = 3.141592;
double p2 = 3.14159265358979323;

printf("p1:\t decimal floating point = %f\n", p1);
printf("\t scientific notation = %e\n", p1);
printf("\t 100 pi = %e\n", p1 * 100);
printf("\t 10 decimal places = %.10f\n", p1);
printf("p2:\t default decimal places = %f\n", p2);
printf("\t 10 decimal places = %.10f\n", p2);
```

Output

```
p1:     decimal floating point = 3.141592
        scientific notation =  3.141592e+00
        100 pi = 3.141592e+02
        10 decimal places = 3.1415920258
```

```
int main(void) {
  int nHrs = 40;
  int nDays = 7;

  float avg = nHrs/nDays;
  printf("%d hours in %d days\n",nHrs,nDays);
  printf("work %.1f hours per day!\n", avg);
  ...
```

main	
nHrs	40
nDays	7
avg	5.0

Output

```
40 hours in 7 days
work 5.0 hours per day!
```

```
int main(void) {
  int nHrs = 40;
  int nDays = 7;

  float avg = nHrs/(float)nDays;
  printf("%d hours in %d days\n",nHrs,nDays);
  printf("work %.1f hours per day!\n", avg);
  ...
```

main	
nHrs	40
nDays	7
avg	5.714285

Output

```
40 hours in 7 days
work 5.7 hours per day!
```

```
struct rect_t {
  int left;
  int bottom;
  int right;
  int top;
};

int main(void) {
  struct rect_t myRect;
  myRect.left = -4;
  myRect.bottom = 1;
  myRect.right = 8;
  myRect.top = 6;

  printf("Bottom left = (%d,%d)\n", myRect.left, myRect.bottom);
  printf("Top right   = (%d,%d)\n", myRect.right, myRect.top);
}
```

Output

Bottom left = (-4, 1)

Other Uses of Typedef

```
typedef unsigned int rgb_t;
rgb_t getRedForPixel(int x, int y)
rgb_t getGreenForPixel(int x, int y)
rgb_t getBlueForPixel(int x, int y)
int main(void) {
  rgb_t red, green, blue;
  //……
  red = getRedForPixel(0,0);
  green = getGreenForPixel(0,0);
  blue = getBlueForPixel(0,0);
  //…
```

What if we used typedef to make rgb_t a type?

Now, we use rgb_t everywhere…

pip value.

Other Uses of Typedef

```
unsigned getRedForPixel(int x, int y)
unsigned getGreenForPixel(int x, int y)
unsigned getBlueForPixel(int x, int y)
int main(void) {
  unsigned red, green, blue;
  //……
  red = getRedForPixel(0,0);
  green = getGreenForPixel(0,0);
  blue = getBlueForPixel(0,0);
  //…
```

What if we want to change to using unsigned char?

Other Uses of Typedef

```
unsigned char getRedForPixel(int x, int y)
unsigned char getGreenForPixel(int x, int y)
unsigned char getBlueForPixel(int x, int y)
int main(void) {
  unsigned char red, green, blue;
  //……
  red = getRedForPixel(0,0);
  green = getGreenForPixel(0,0);
  blue = getBlueForPixel(0,0);
  //…
```

We need to change every place we wrote the type

Other Uses of Typedef

```
typedef unsigned int rgb_t;
rgb_t getRedForPixel(int x, int y)
rgb_t getGreenForPixel(int x, int y)
rgb_t getBlueForPixel(int x, int y)
int main(void) {
  rgb_t red, green, blue;
  //……
  red = getRedForPixel(0,0);
  green = getGreenForPixel(0,0);
  blue = getBlueForPixel(0,0);
  //…
```

What if we used typedef to make rgb_t a type?

Now, we use rgb_t everywhere…

Other Uses of Typedef

```
typedef unsigned char rgb_t;
rgb_t getRedForPixel(int x, int y)
rgb_t getGreenForPixel(int x, int y)
rgb_t getBlueForPixel(int x, int y)
int main(void) {
  rgb_t red, green, blue;
  //……
  red = getRedForPixel(0,0);
  green = getGreenForPixel(0,0);
  blue = getBlueForPixel(0,0);
  //…
```

Only change: what we typedef rgb_t to be!

```c
enum threat_level_t {
  LOW,
  GUARDED,
  ELEVATED,
  HIGH,
  SEVERE
};

void printThreat(enum threat_level_t threat) {/* omitted */}

void printShoes(enum threat_level_t myThreat) {/* omitted */}

int main(void) {
  enum threat_level_t myThreat = HIGH;

  printf("Current threat level is:\n");
  printThreat(myThreat);
  printShoes(myThreat);
  return 0;
}
```

```
enum threat_level_t {
   LOW,          0
   GUARDED,      1
   ELEVATED,     2
   HIGH,         3
   SEVERE        4
};
```

```
enum threat_level_t { LOW, GUARDED, ELEVATED, HIGH, SEVERE };

void printThreat(enum threat_level_t threat){
  switch(threat) {
    case LOW:
      printf("Green/Low.\n");
      break;
    case GUARDED:
      printf("Blue/Guarded.\n");
      break;
    case ELEVATED:
      printf("Yellow/Elevated.\n");
      break;
    case HIGH:
      printf("Orange/High.\n");
      break;
    case SEVERE:
      printf("Red/Severe.\n");
      break;
  }
}
```

```c
enum threat_level_t {
  LOW,
  GUARDED,
  ELEVATED,
  HIGH,
  SEVERE
};

int main(void) {
  enum threat_level_t myThreat = HIGH;

  printf("Current threat level is:\n");
  printThreat(myThreat);
  printShoes(myThreat);
  return 0;
}
```

Output

Current threat level is:

```
enum threat_level_t { LOW, GUARDED, ELEVATED, HIGH, SEVERE };

void printThreat(enum threat_level_t threat){
  switch(threat) {
    case LOW:
      printf("Green/Low.\n");
      break;
    case GUARDED:
      printf("Blue/Guarded.\n");
      break;
    case ELEVATED:
      printf("Yellow/Elevated.\n");
      break;
    case HIGH:
      printf("Orange/High.\n");
      break;
    case SEVERE:
      printf("Red/Severe.\n");
      break;
  }
}
```

Output

Current threat level is:
Orange/High.

```
enum threat_level_t { LOW, GUARDED, ELEVATED, HIGH, SEVERE };

void printShoes(enum threat_level_t currThreat) {
    if (currThreat >= ELEVATED) {
        printf("Please take off your shoes.\n");
    }
    else {
        printf("Please leave your shoes on.\n");
    }
}
```

Output

Current threat level is:
Orange/High.

Introduction to Sorting

Final project in this Course: write a **sorting** algorithm

Put data in ascending order

Why sorting?

Important/ubiquitous algorithm in CS

Importance of Sorting