Assignment-2

Objective: Execution of python programs

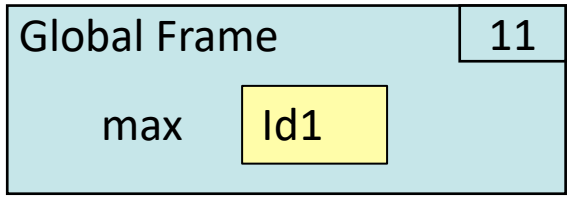
Date of submission: 17 Jan 2023

Submitted by Group: Shravan , Ashutosh and Vidya

Guided by: Prof. Sonika Thakral ma'am

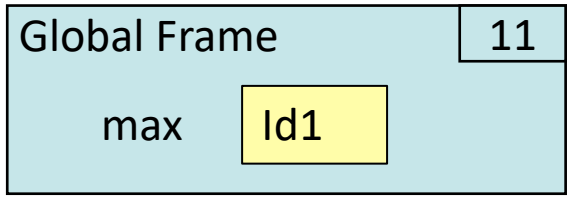
Part A

```
1  def max(a,b,c):
2      """Returns: maximum of a, b, and c
3
4      Precondition: a, b, c are numbers"""
5      m = a
6      if (c > b and c > a):
7          m = c
8      elif b > a:
9          m = b
10     return m
11 d = max(1,2,3)
```

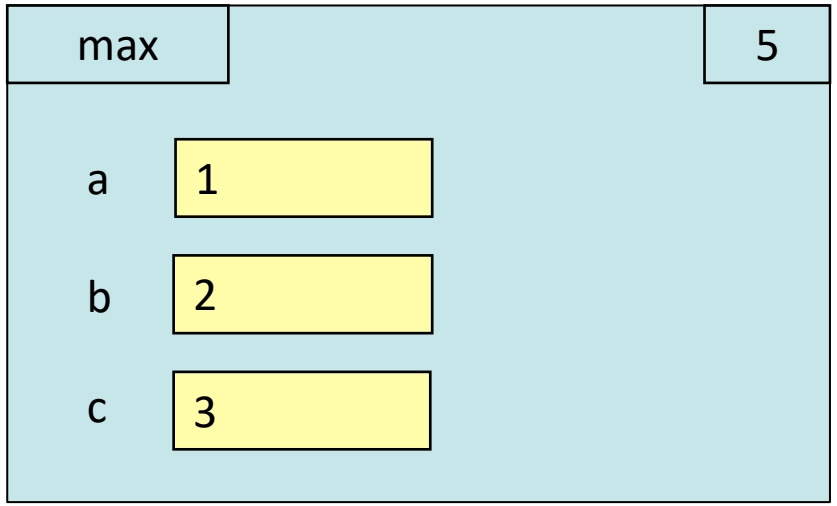


Part A

```
1  def max(a,b,c):
2      """Returns: maximum of a, b, and c
3
4      Precondition: a, b, c are numbers"""
5      m = a
6      if (c > b and c > a):
7          m = c
8      elif b > a:
9          m = b
10     return m
11 d = max(1,2,3)
```

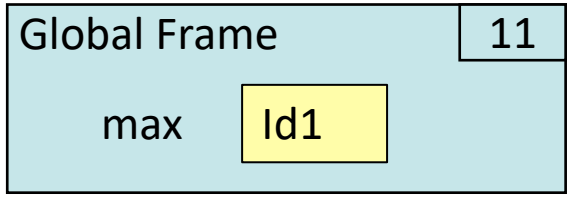


Call Stack

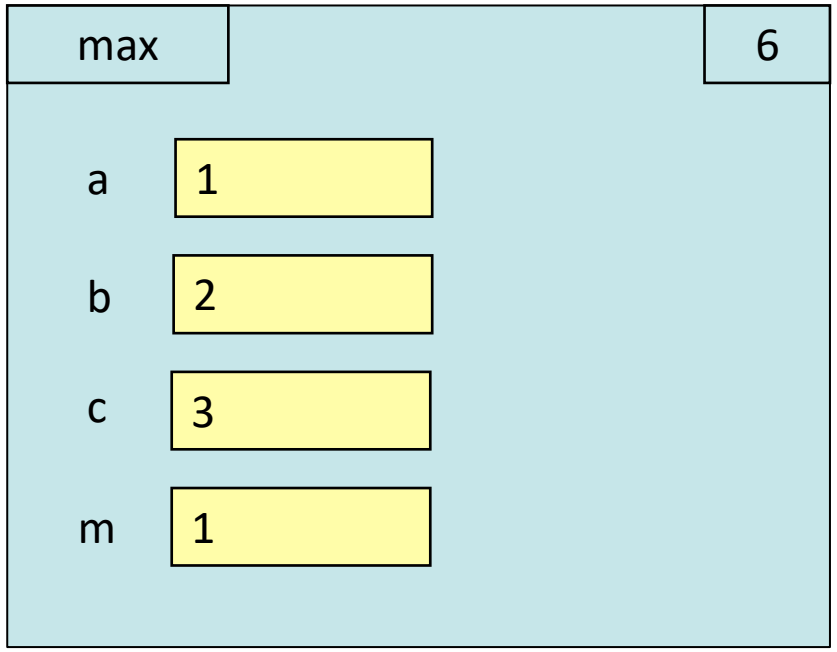


Part A

```
1  def max(a,b,c):
2      """Returns: maximum of a, b, and c
3
4      Precondition: a, b, c are numbers"""
5      m = a
6      if (c > b and c > a):
7          m = c
8      elif b > a:
9          m = b
10     return m
11 d = max(1,2,3)
```

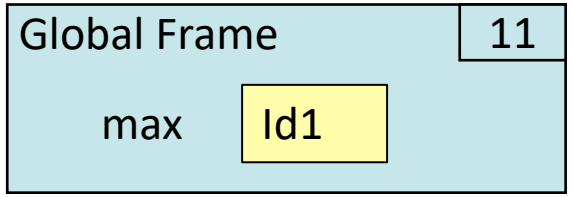


Call Stack

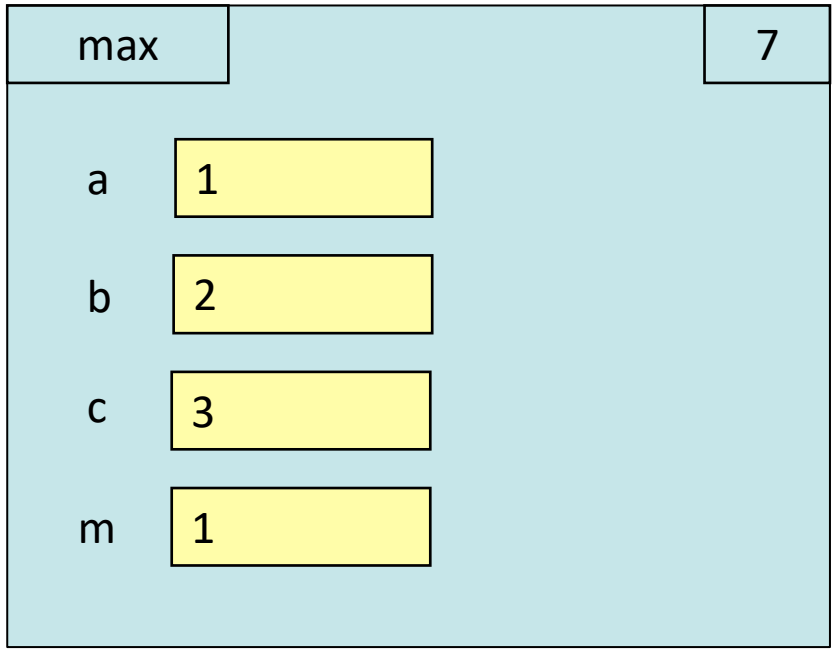


Part A

```
1  def max(a,b,c):
2      """Returns: maximum of a, b, and c
3
4      Precondition: a, b, c are numbers"""
5      m = a
6      if (c > b and c > a):
7          m = c
8      elif b > a:
9          m = b
10     return m
11 d = max(1,2,3)
```

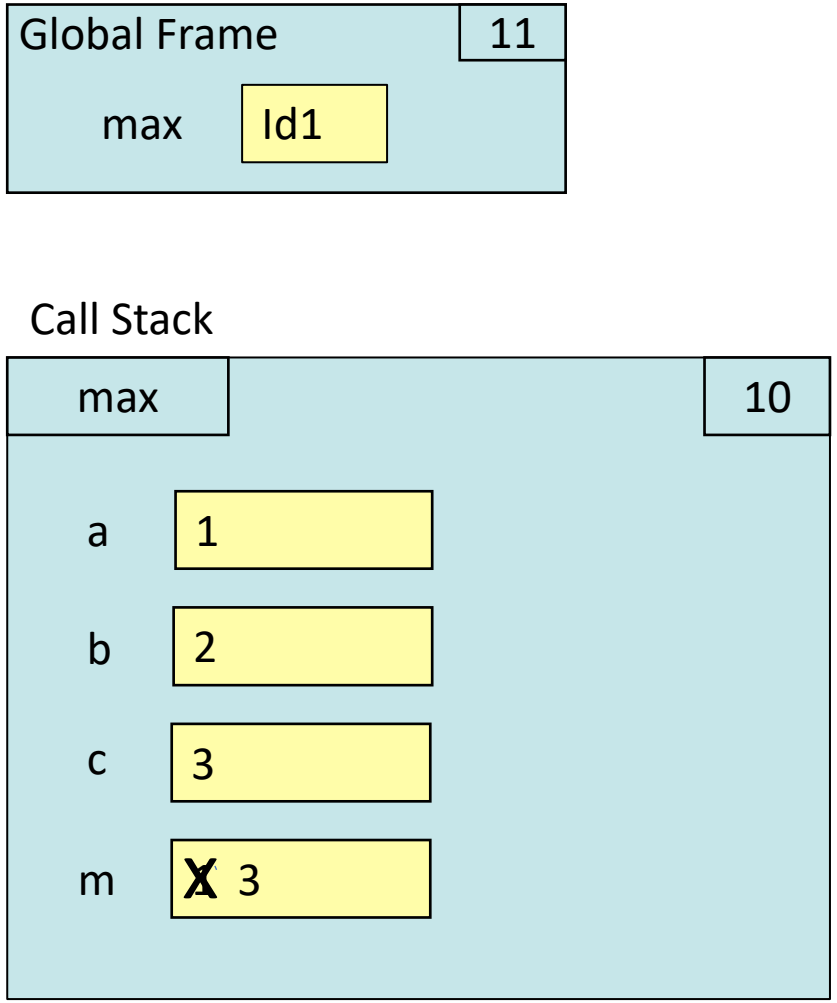


Call Stack



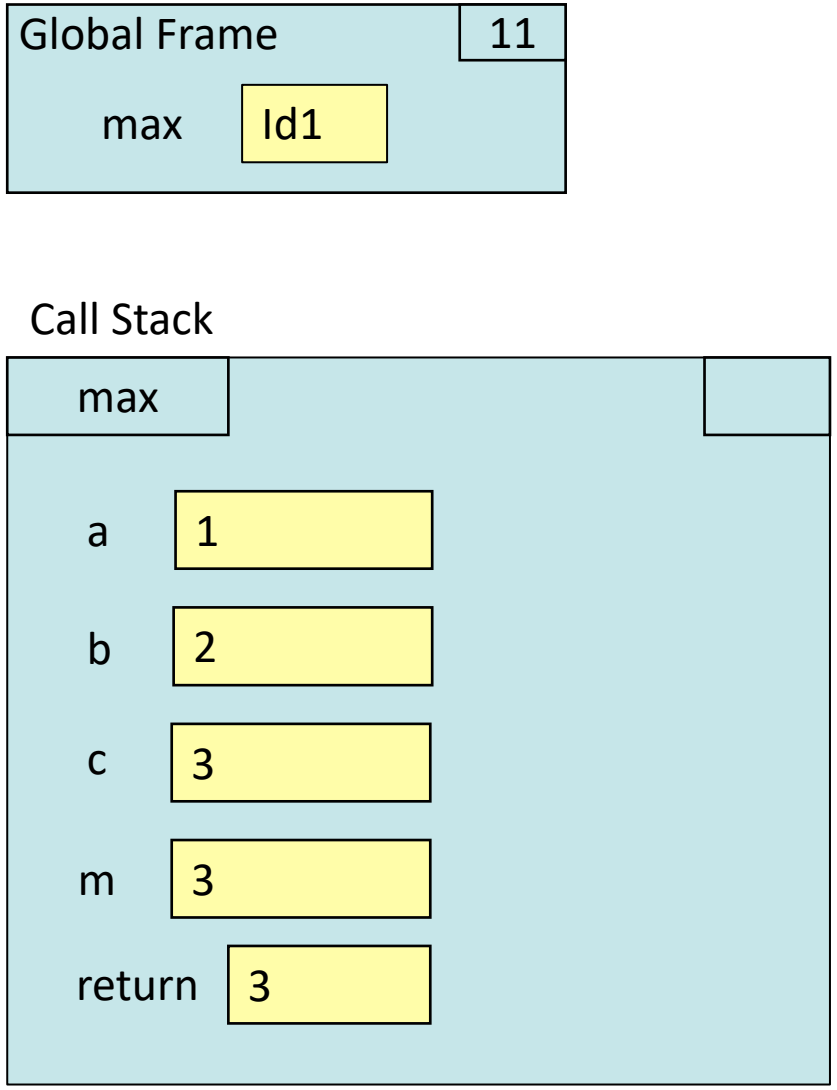
Part A

```
1  def max(a,b,c):
2      """Returns: maximum of a, b, and c
3
4      Precondition: a, b, c are numbers"""
5      m = a
6      if (c > b and c > a):
7          m = c
8      elif b > a:
9          m = b
10     return m
11 d = max(1,2,3)
```



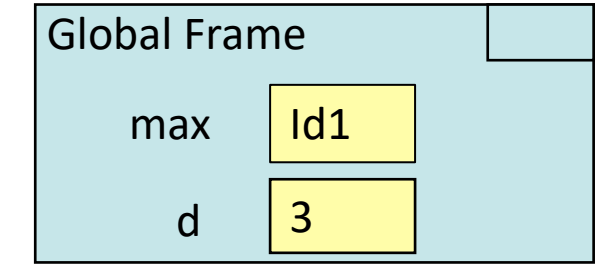
Part A

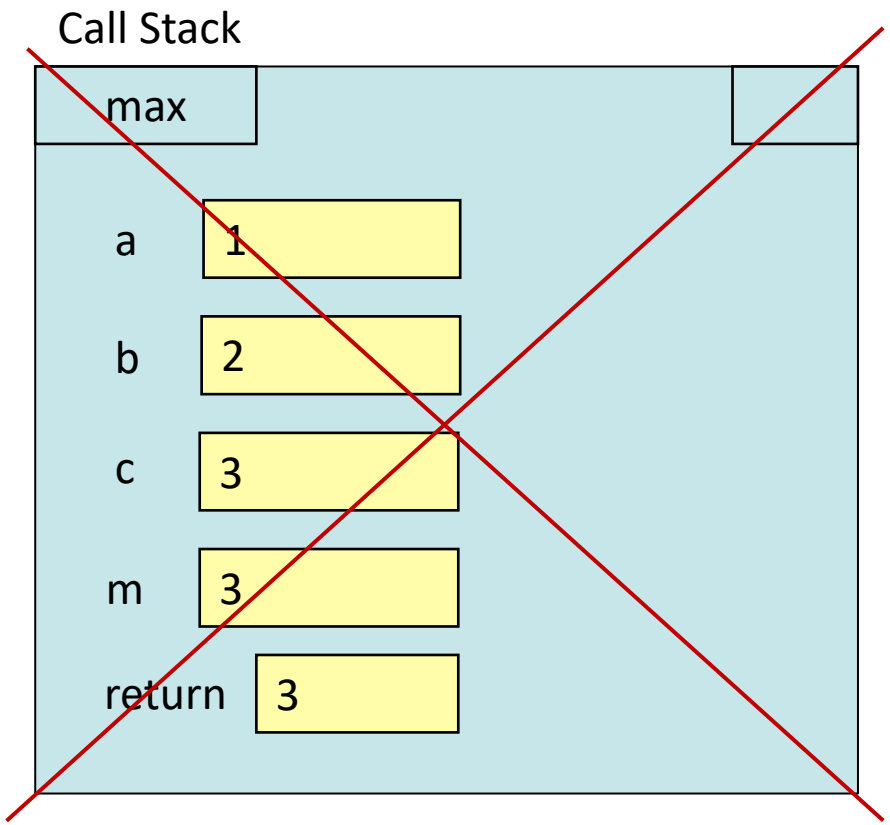
```
1  def max(a,b,c):
2      """Returns: maximum of a, b, and c
3
4      Precondition: a, b, c are numbers"""
5      m = a
6      if (c > b and c > a):
7          m = c
8      elif b > a:
9          m = b
10     return m
11 d = max(1,2,3)
```



Part A

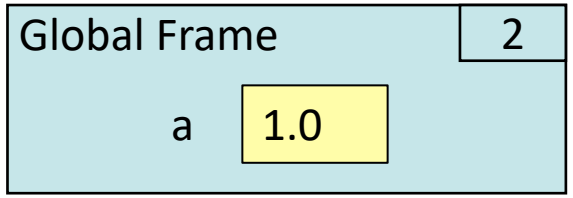
```
1  def max(a,b,c):
2      """Returns: maximum of a, b, and c
3
4      Precondition: a, b, c are numbers"""
5      m = a
6      if (c > b and c > a):
7          m = c
8      elif b > a:
9          m = b
10     return m
11 d = max(1,2,3)
```





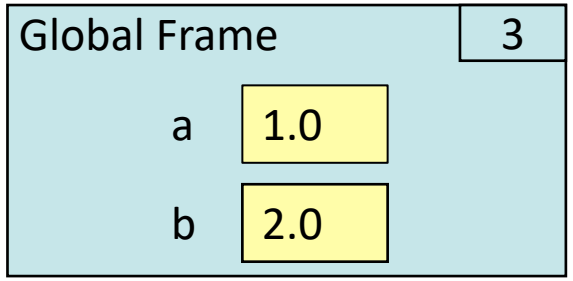
Part B

```
1  a = 1.0
2  b = 2.0
3  c = 3.0
4  def max(a,b,c):
5      """Returns: maximum of a, b, and c
6
7      Precondition: a, b, c are numbers"""
8      m = a
9      if (c > b and c > a):
10         m = c
11     elif b > a:
12         m = b
13     return m
14 e = max(c,b,a)
```



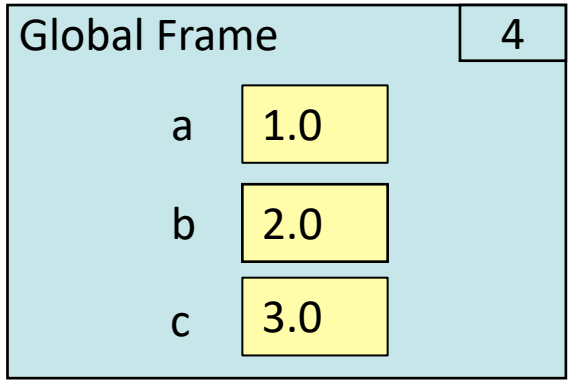
Part B

```
1  a = 1.0
2  b = 2.0
3  c = 3.0
4  def max(a,b,c):
5      """Returns: maximum of a, b, and c
6
7      Precondition: a, b, c are numbers"""
8      m = a
9      if (c > b and c > a):
10         m = c
11     elif b > a:
12         m = b
13     return m
14 e = max(c,b,a)
```



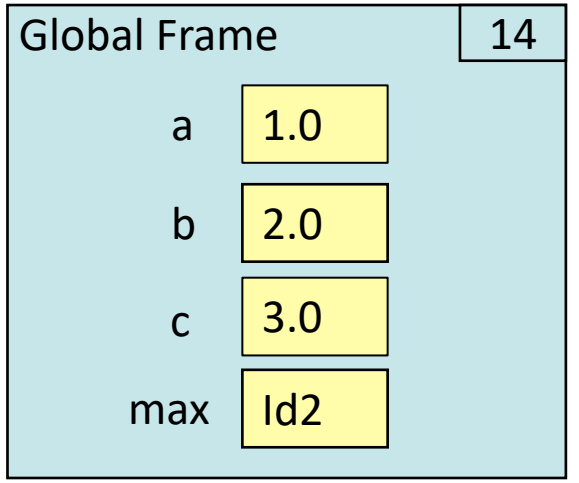
Part B

```
1  a = 1.0
2  b = 2.0
3  c = 3.0
4  def max(a,b,c):
5      """Returns: maximum of a, b, and c
6
7      Precondition: a, b, c are numbers"""
8      m = a
9      if (c > b and c > a):
10         m = c
11     elif b > a:
12         m = b
13     return m
14 e = max(c,b,a)
```



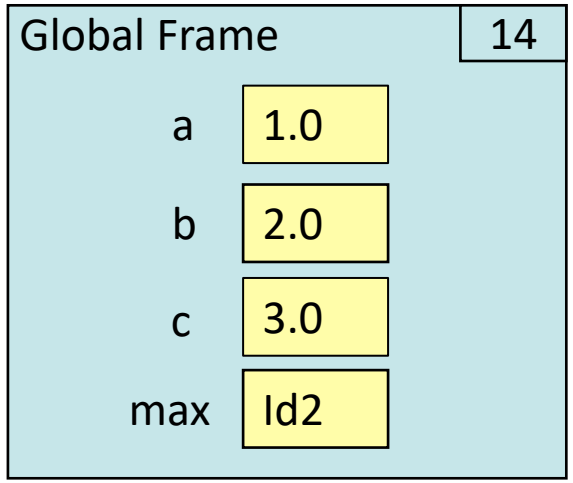
Part B

```
1  a = 1.0
2  b = 2.0
3  c = 3.0
4  def max(a,b,c):
5      """Returns: maximum of a, b, and c
6
7      Precondition: a, b, c are numbers"""
8      m = a
9      if (c > b and c > a):
10         m = c
11     elif b > a:
12         m = b
13     return m
14 e = max(c,b,a)
```

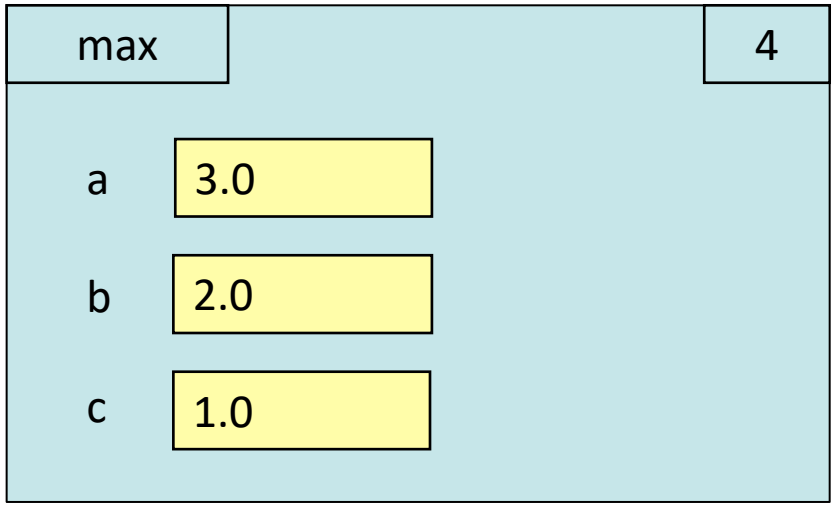


Part B

```
1  a = 1.0
2  b = 2.0
3  c = 3.0
4  def max(a,b,c):
5      """Returns: maximum of a, b, and c
6
7      Precondition: a, b, c are numbers"""
8      m = a
9      if (c > b and c > a):
10         m = c
11     elif b > a:
12         m = b
13     return m
14 e = max(c,b,a)
```

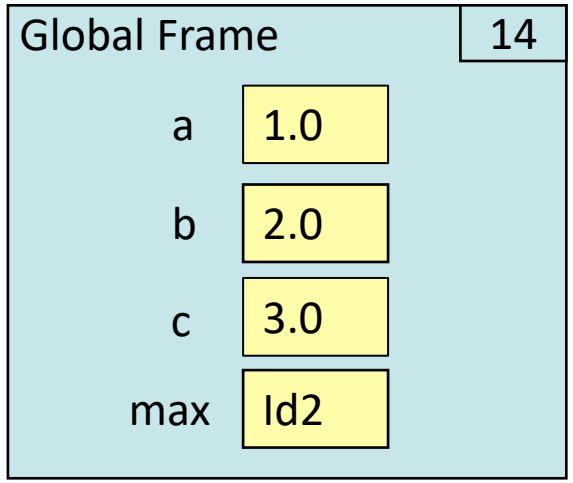


Call Stack

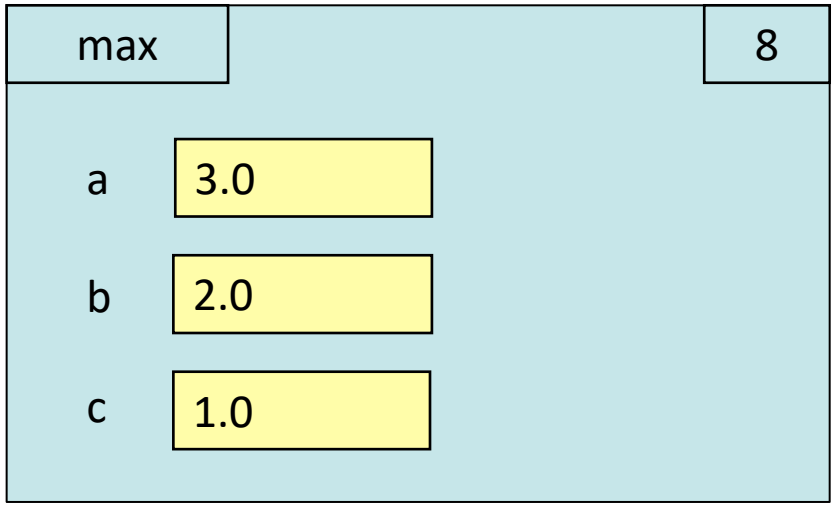


Part B

```
1  a = 1.0
2  b = 2.0
3  c = 3.0
4  def max(a,b,c):
5      """Returns: maximum of a, b, and c
6
7      Precondition: a, b, c are numbers"""
8      m = a
9      if (c > b and c > a):
10         m = c
11     elif b > a:
12         m = b
13     return m
14 e = max(c,b,a)
```

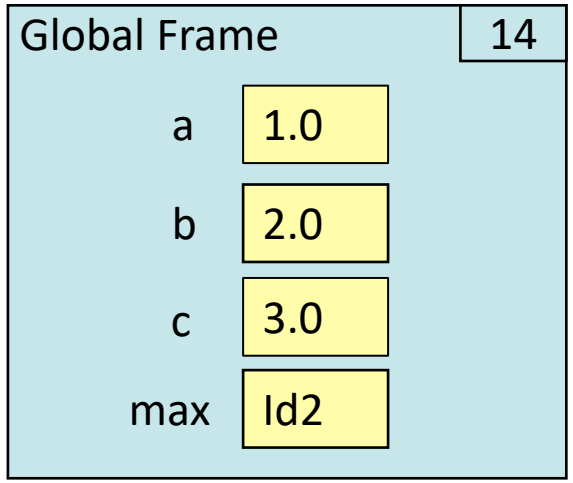


Call Stack

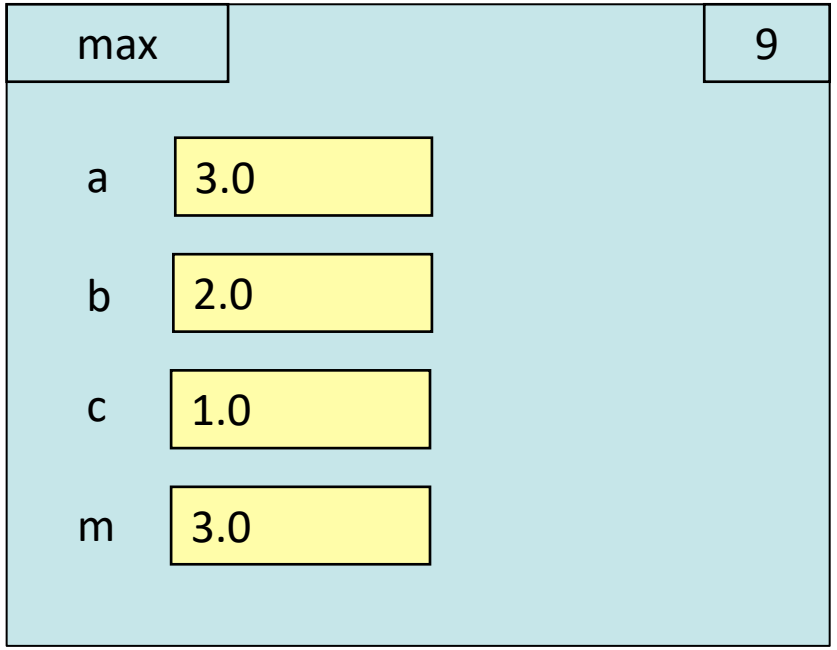


Part B

```
1  a = 1.0
2  b = 2.0
3  c = 3.0
4  def max(a,b,c):
5      """Returns: maximum of a, b, and c
6
7      Precondition: a, b, c are numbers"""
8      m = a
9      if (c > b and c > a):
10         m = c
11     elif b > a:
12         m = b
13     return m
14 e = max(c,b,a)
```

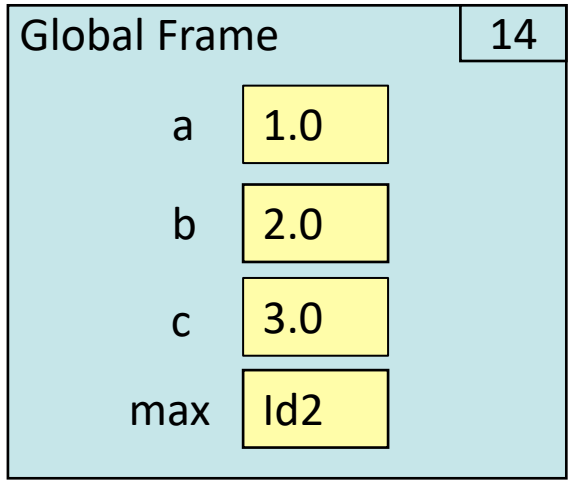


Call Stack

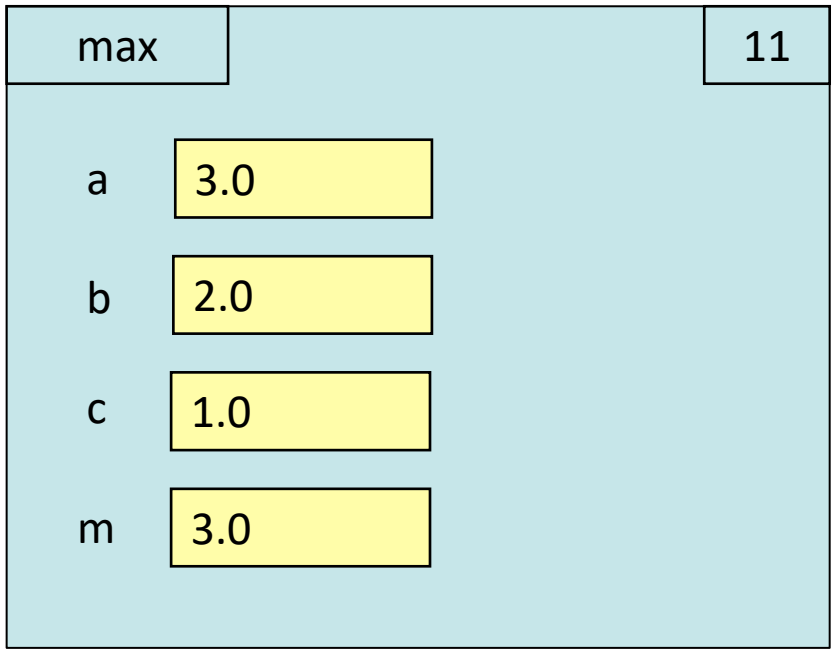


Part B

```
1  a = 1.0
2  b = 2.0
3  c = 3.0
4  def max(a,b,c):
5      """Returns: maximum of a, b, and c
6
7      Precondition: a, b, c are numbers"""
8      m = a
9      if (c > b and c > a):
10         m = c
11     elif b > a:
12         m = b
13     return m
14 e = max(c,b,a)
```

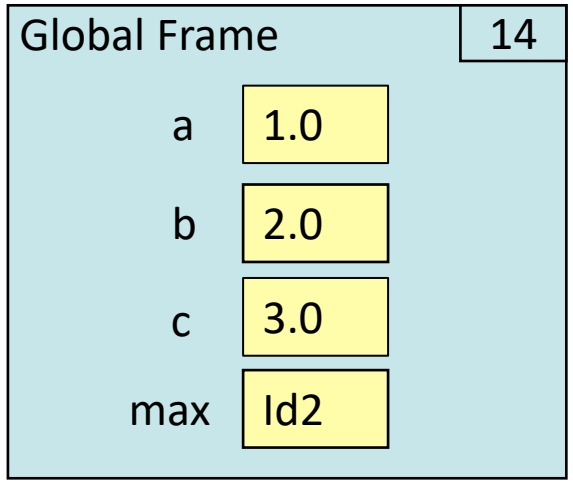


Call Stack

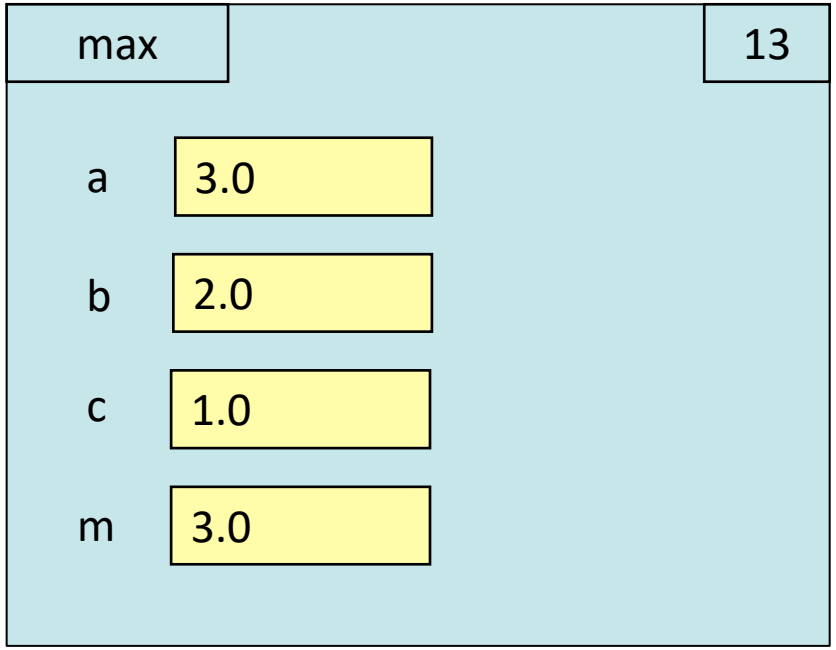


Part B

```
1  a = 1.0
2  b = 2.0
3  c = 3.0
4  def max(a,b,c):
5      """Returns: maximum of a, b, and c
6
7      Precondition: a, b, c are numbers"""
8      m = a
9      if (c > b and c > a):
10         m = c
11     elif b > a:
12         m = b
13     return m
14 e = max(c,b,a)
```

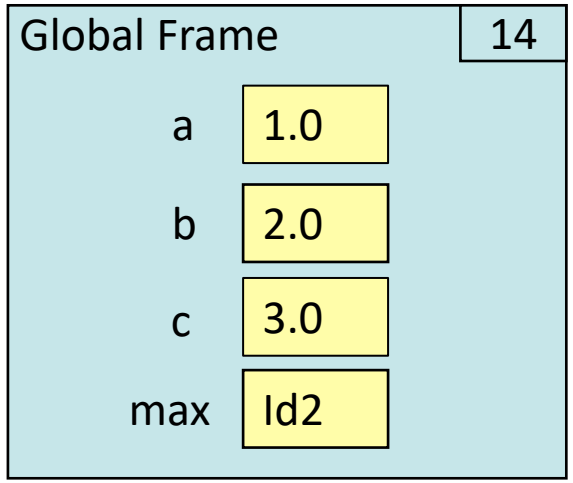


Call Stack

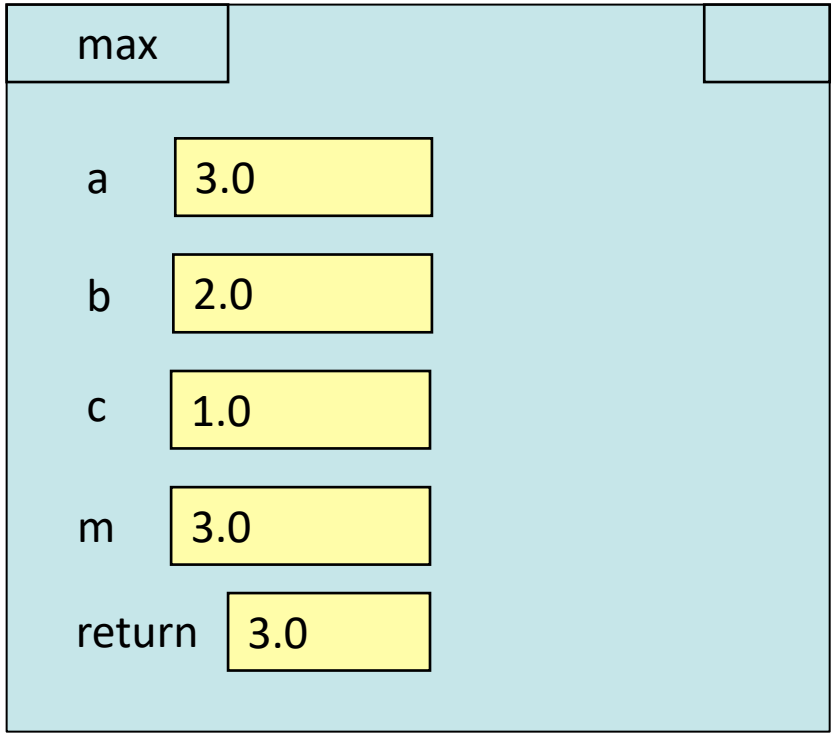


Part B

```
1  a = 1.0
2  b = 2.0
3  c = 3.0
4  def max(a,b,c):
5      """Returns: maximum of a, b, and c
6
7      Precondition: a, b, c are numbers"""
8      m = a
9      if (c > b and c > a):
10         m = c
11     elif b > a:
12         m = b
13     return m
14 e = max(c,b,a)
```

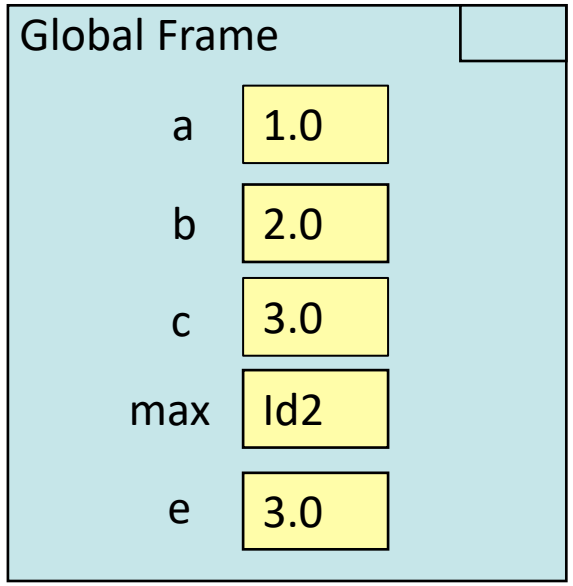


Call Stack

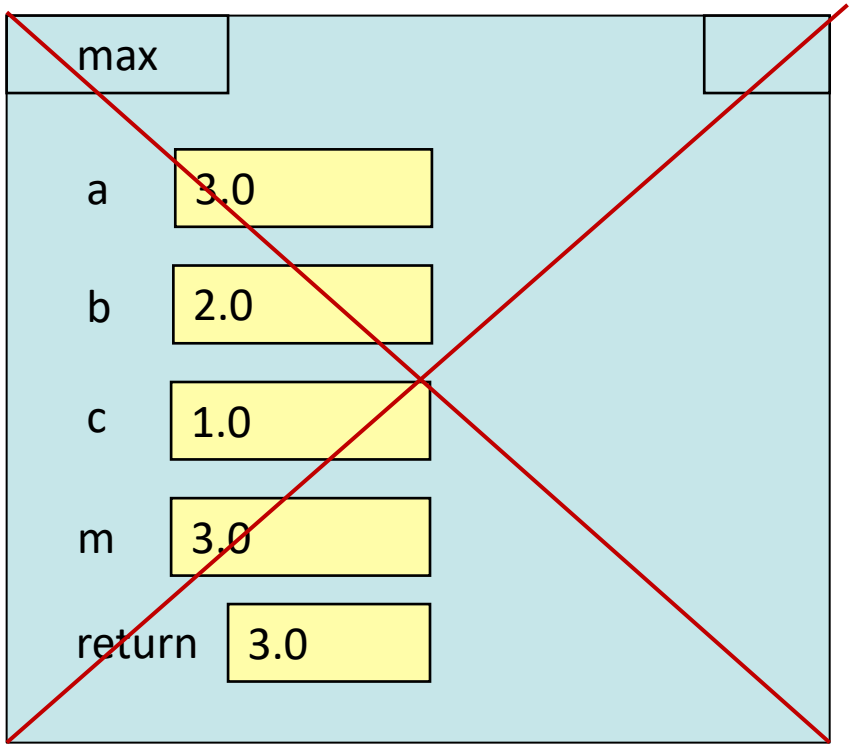


Part B

```
1  a = 1.0
2  b = 2.0
3  c = 3.0
4  def max(a,b,c):
5      """Returns: maximum of a, b, and c
6
7      Precondition: a, b, c are numbers"""
8      m = a
9      if (c > b and c > a):
10         m = c
11     elif b > a:
12         m = b
13     return m
14 e = max(c,b,a)
```

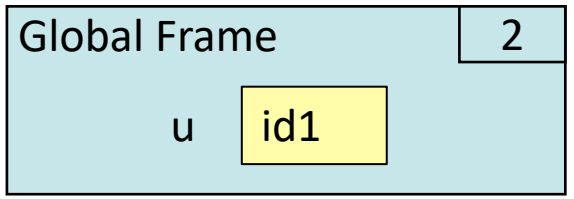


Call Stack

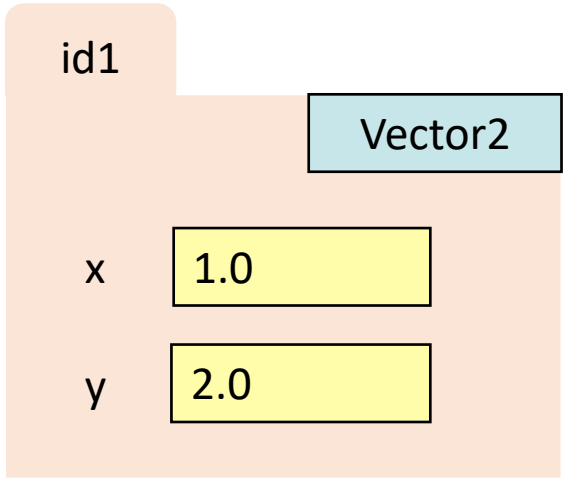


Part C

```
1  u = Vector2(1.0,2.0)
2  v = Vector2(3.0,4.0)
3  def perp(v):
4      """Modifies v to be perpendicular to the original
5
6      Precondition: v is a Vector2 object"""
7      tmp = v.y
8      v.y = v.x
9      v.x = -tmp
10 perp(u)
```

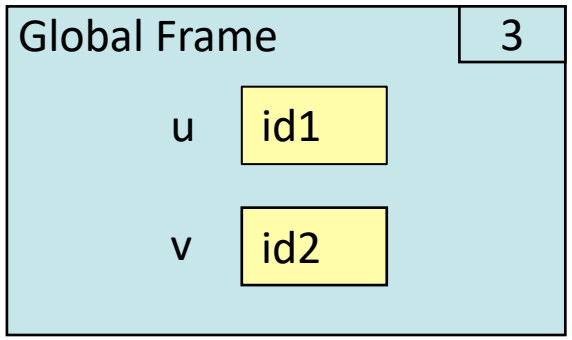


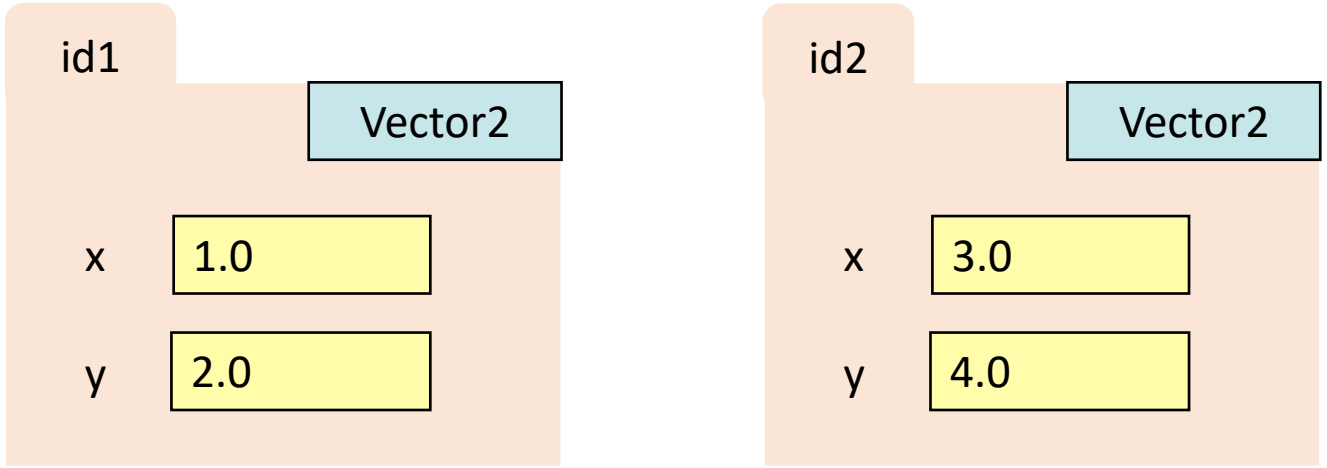
Heap Space



Part C

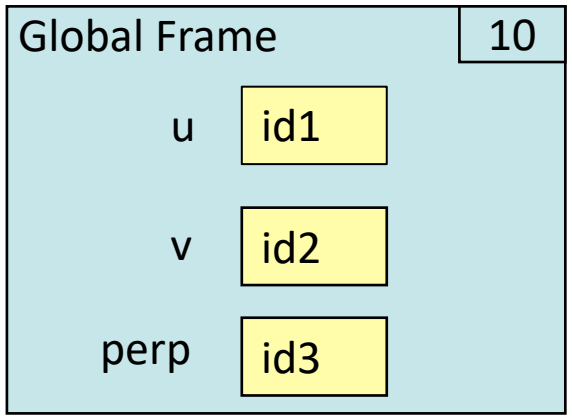
```
1  u = Vector2(1.0,2.0)
2  v = Vector2(3.0,4.0)
3  def perp(v):
4      """Modifies v to be perpendicular to the original
5
6      Precondition: v is a Vector2 object"""
7      tmp = v.y
8      v.y = v.x
9      v.x = -tmp
10 perp(u)
```

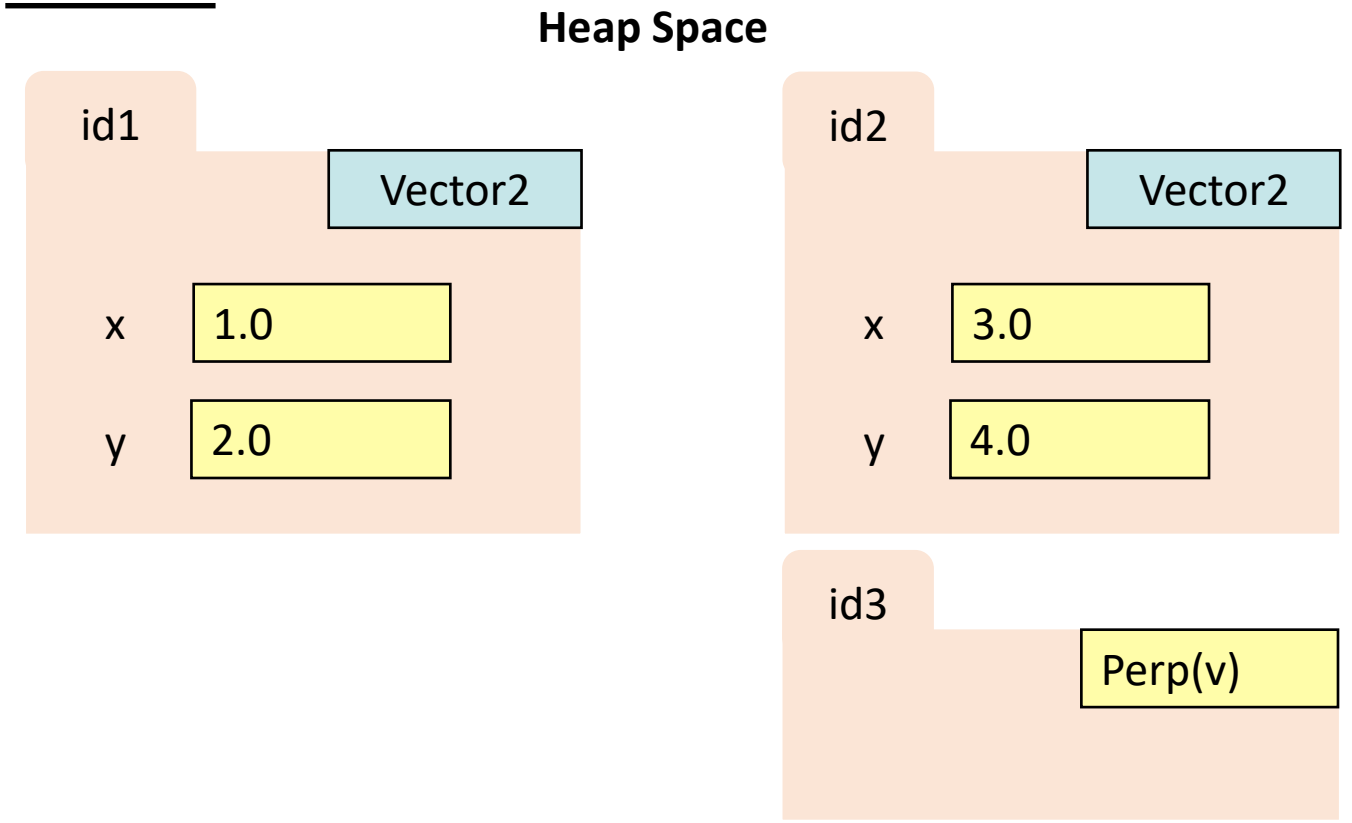




Part C

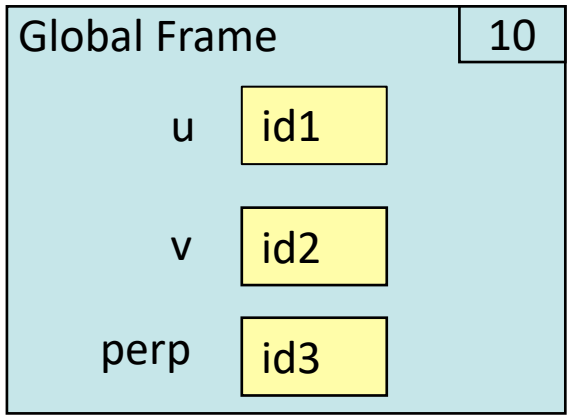
```
1  u = Vector2(1.0,2.0)
2  v = Vector2(3.0,4.0)
3  def perp(v):
4      """Modifies v to be perpendicular to the original
5
6      Precondition: v is a Vector2 object"""
7      tmp = v.y
8      v.y = v.x
9      v.x = -tmp
10 perp(u)
```



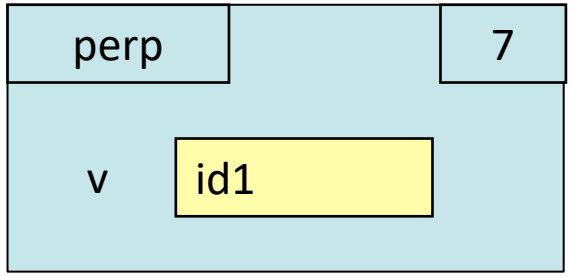


Part C

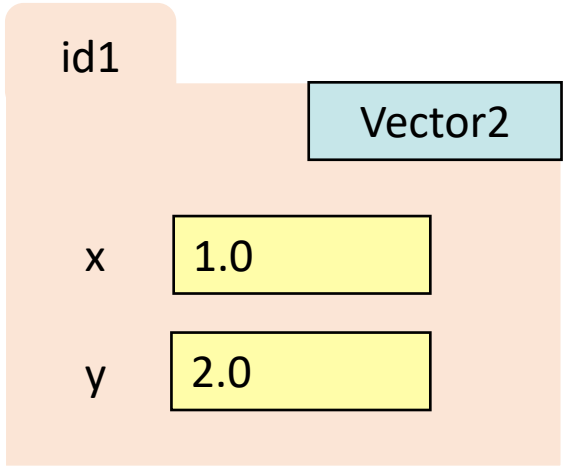
```
1  u = Vector2(1.0,2.0)
2  v = Vector2(3.0,4.0)
3  def perp(v):
4      """Modifies v to be perpendicular to the original
5
6      Precondition: v is a Vector2 object"""
7      tmp = v.y
8      v.y = v.x
9      v.x = -tmp
10 perp(u)
```

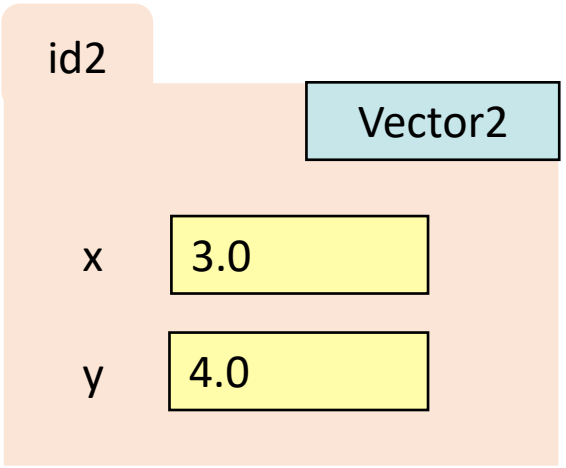


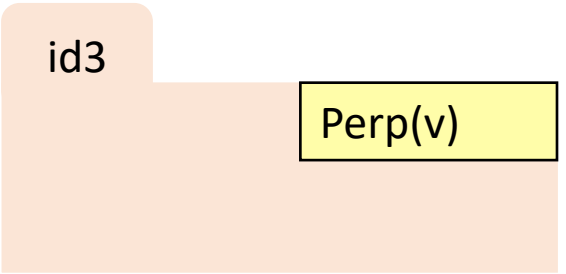
Call Stack



Heap Space

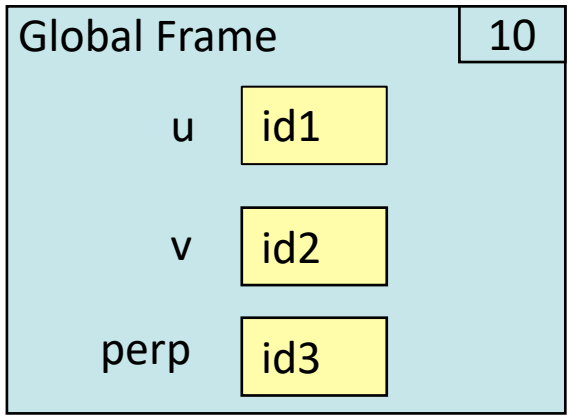




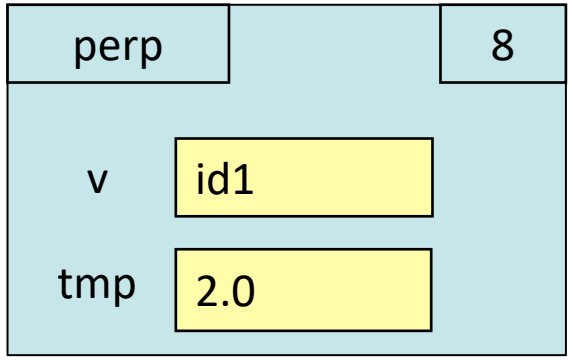


Part C

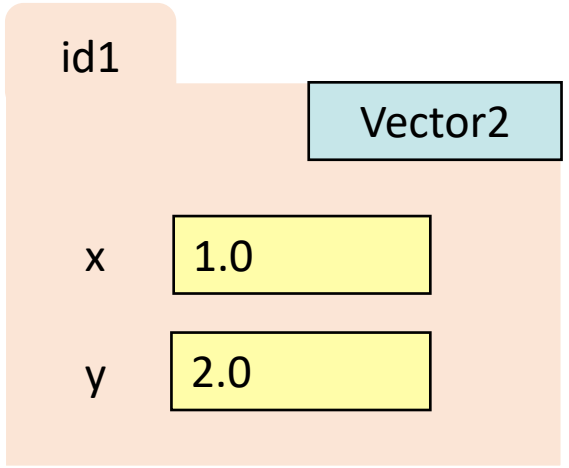
```
1  u = Vector2(1.0,2.0)
2  v = Vector2(3.0,4.0)
3  def perp(v):
4      """Modifies v to be perpendicular to the original
5
6      Precondition: v is a Vector2 object"""
7      tmp = v.y
8      v.y = v.x
9      v.x = -tmp
10 perp(u)
```

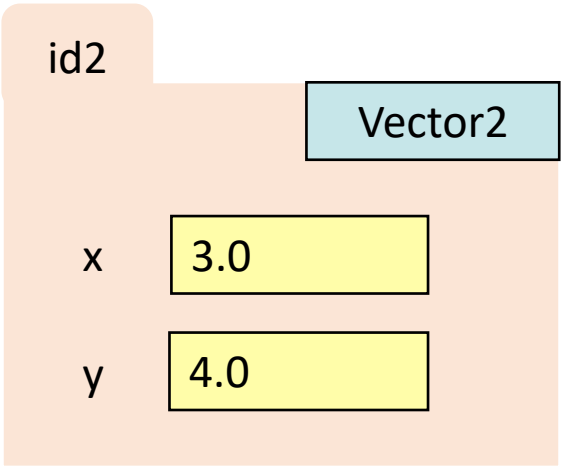


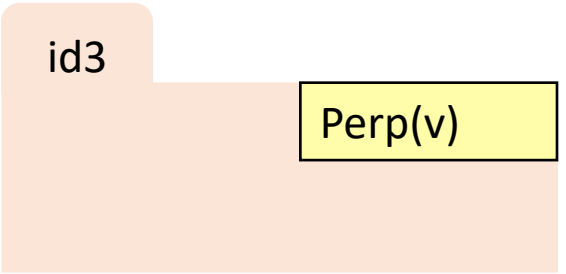
Call Stack



Heap Space







Part C

```
1  u = Vector2(1.0,2.0)
2  v = Vector2(3.0,4.0)
3  def perp(v):
4      """Modifies v to be perpendicular to the original
5
6      Precondition: v is a Vector2 object"""
7      tmp = v.y
8      v.y = v.x
9      v.x = -tmp
10 perp(u)
```

Heap Space

id1

Vector2

x 1.0

y ~~2.0~~ 1.0

id2

Vector2

x 3.0

y 4.0

id3

Perp(v)

Global Frame 10

u id1

v id2

perp id3

Call Stack

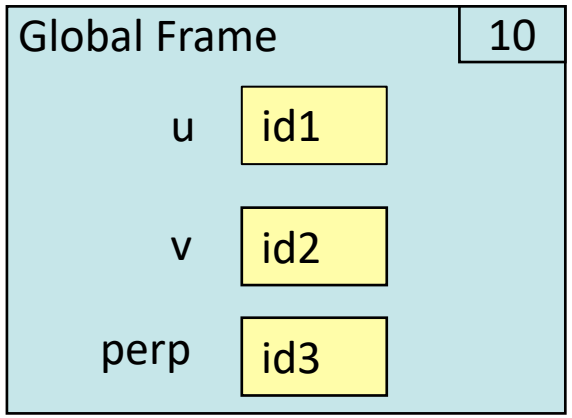
perp 9

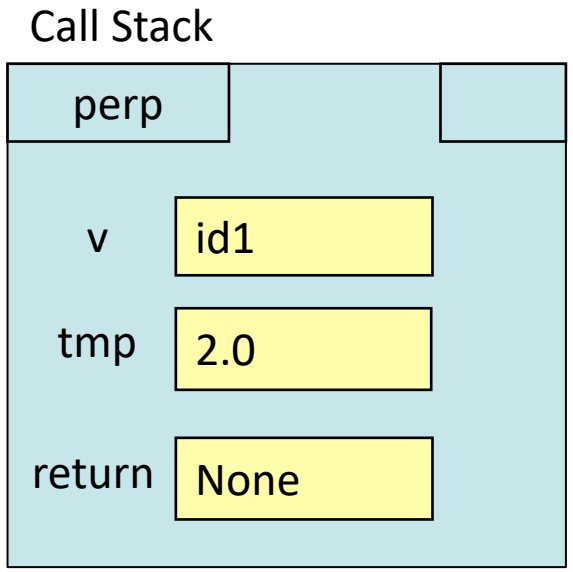
v id1

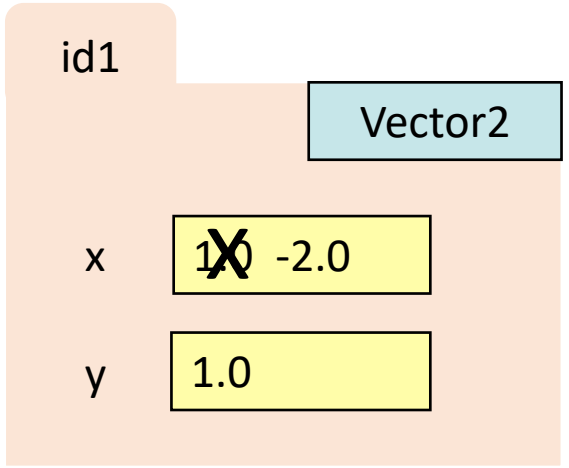
tmp 2.0

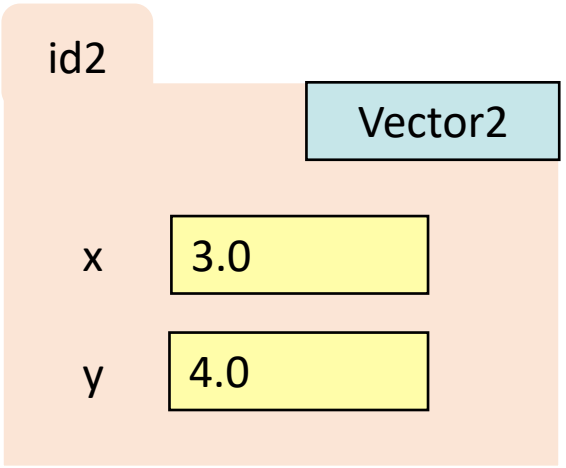
Part C

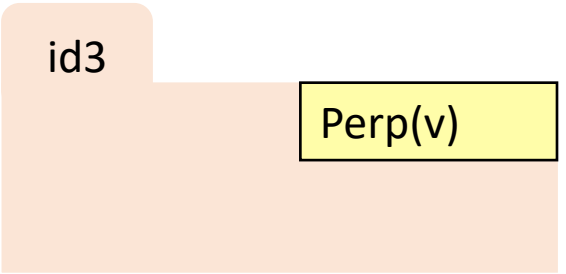
```
1  u = Vector2(1.0,2.0)
2  v = Vector2(3.0,4.0)
3  def perp(v):
4      """Modifies v to be perpendicular to the original
5
6      Precondition: v is a Vector2 object"""
7      tmp = v.y
8      v.y = v.x
9      v.x = -tmp
10 perp(u)
```





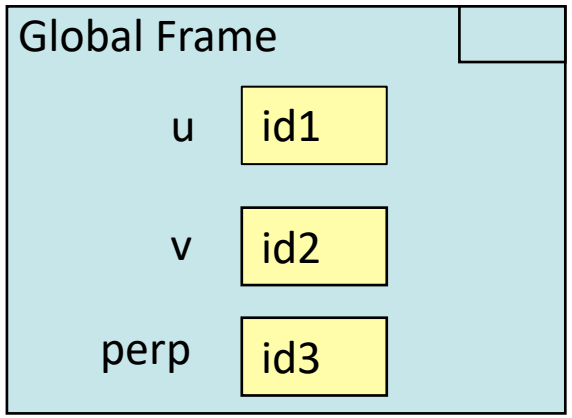


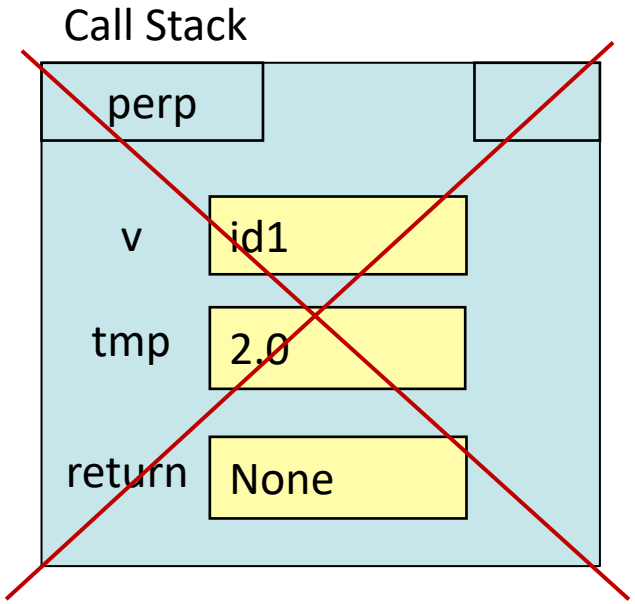


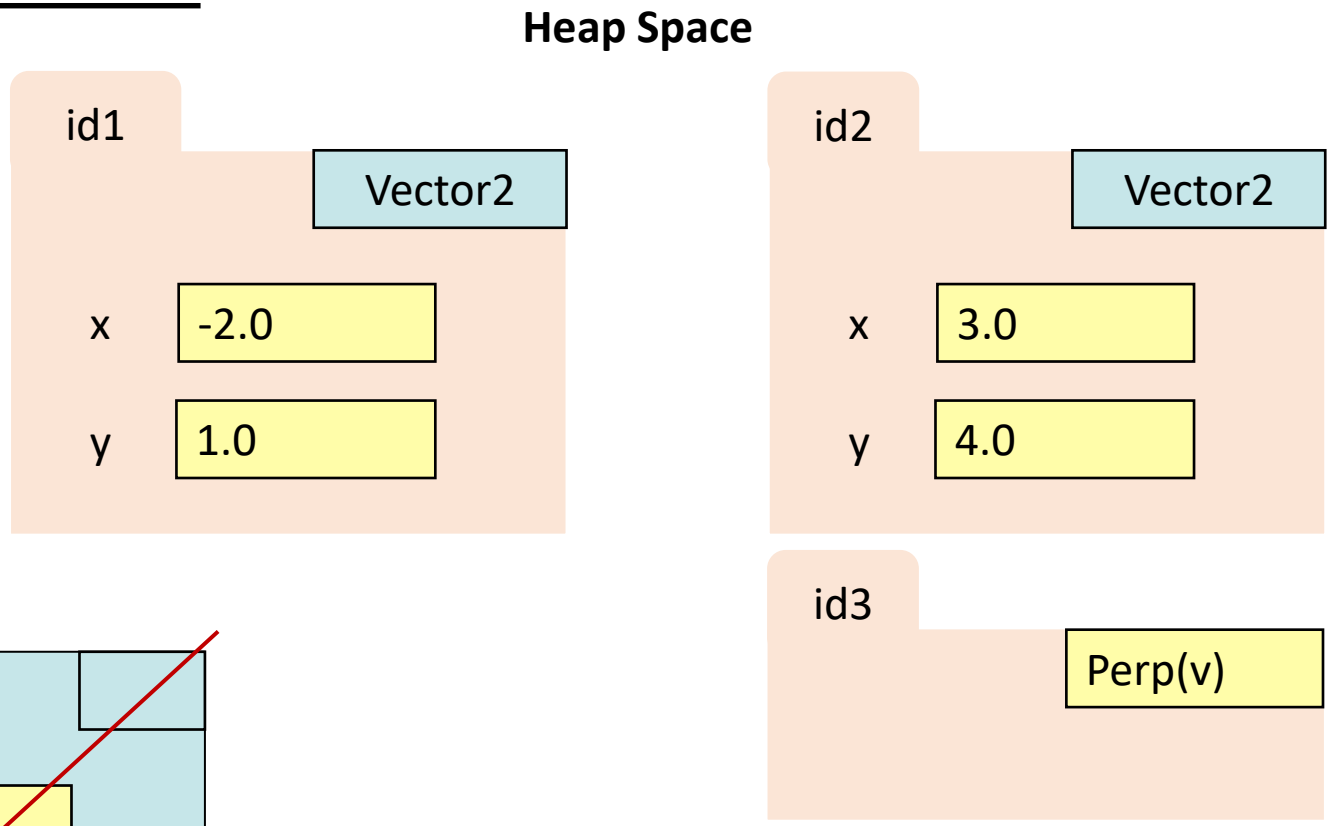


Part C

```
1  u = Vector2(1.0,2.0)
2  v = Vector2(3.0,4.0)
3  def perp(v):
4      """Modifies v to be perpendicular to the original
5
6      Precondition: v is a Vector2 object"""
7      tmp = v.y
8      v.y = v.x
9      v.x = -tmp
10 perp(u)
```

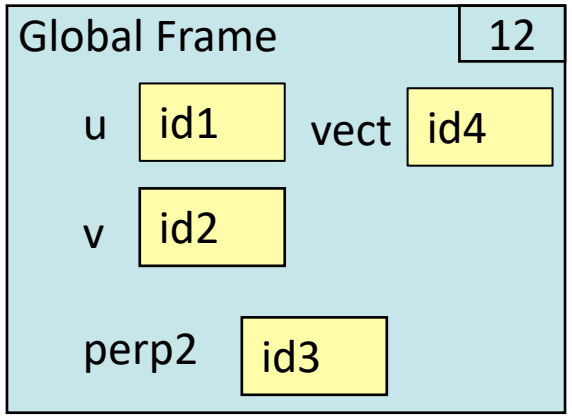




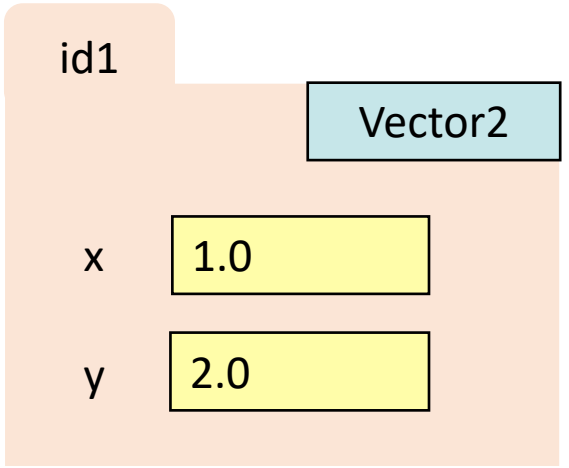


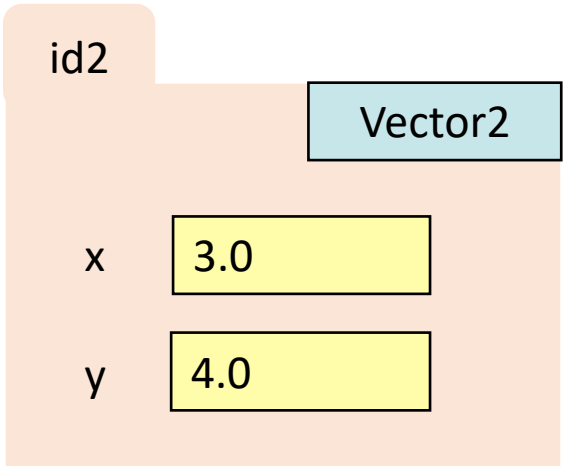
Part D

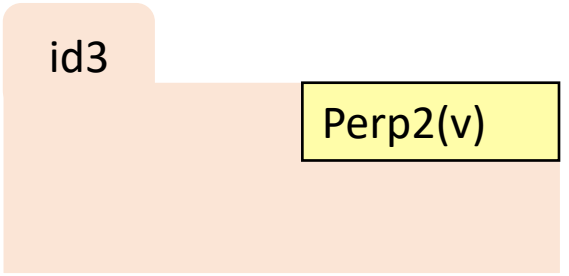
```
1  Import vect
2  u = Vector2(1.0,2.0)
3  v = Vector2(3.0,4.0)
4  def perp2(v):
5      """Returns a vector perpendicular to v
6
7      Precondition: v is a Vector2 object"""
8      u = vect.Vector2(0.0,0.0)
9      u.y = v.x
10     u.x = -v.y
11     return u
12 v = perp2(u)
```

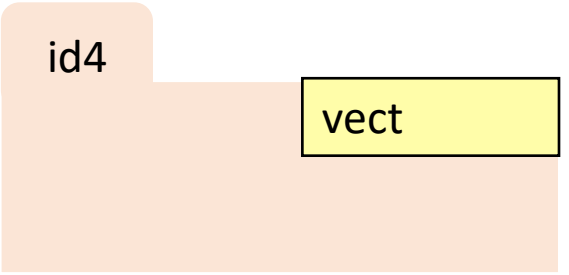


Heap Space



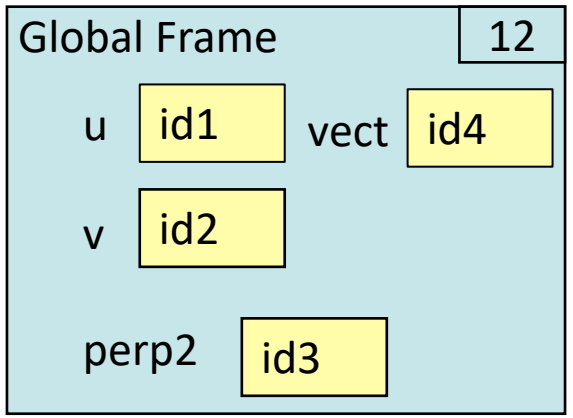


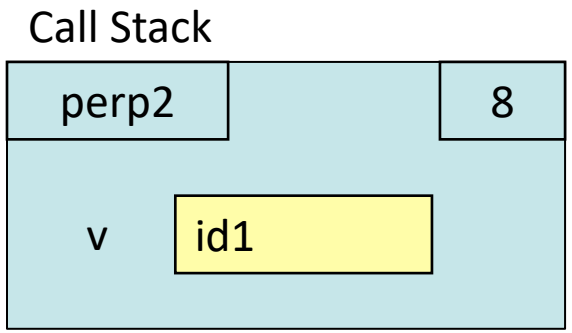




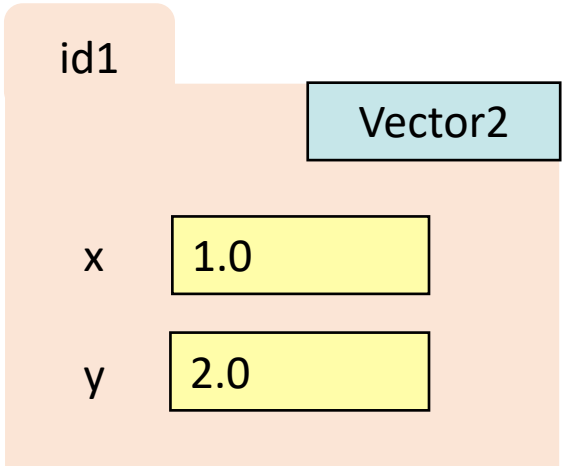
Part D

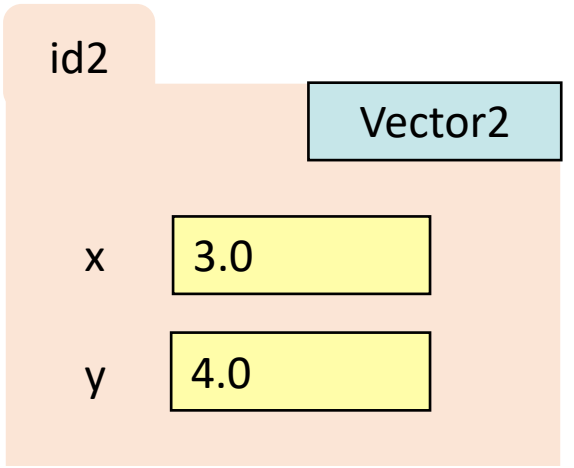
```
1  Import vect
2  u = Vector2(1.0,2.0)
3  v = Vector2(3.0,4.0)
4  def perp2(v):
5      """Returns a vector perpendicular to v
6
7      Precondition: v is a Vector2 object"""
8      u = vect.Vector2(0.0,0.0)
9      u.y = v.x
10     u.x = -v.y
11     return u
12 v = perp2(u)
```

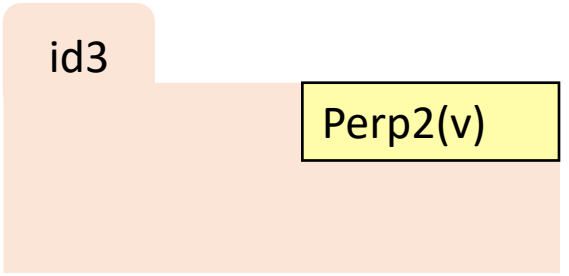


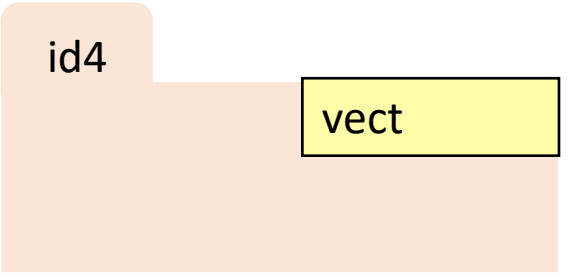


Heap Space



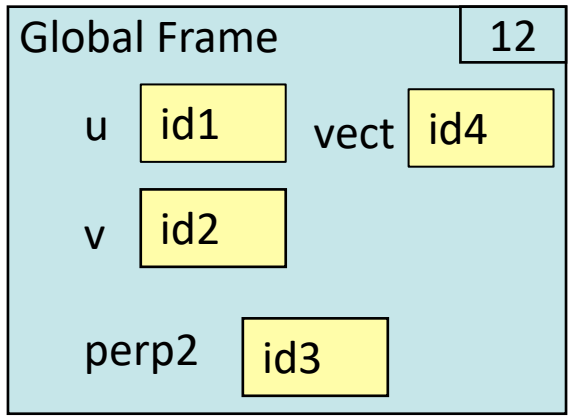


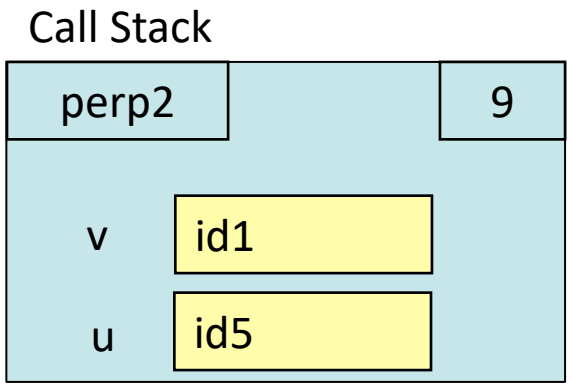


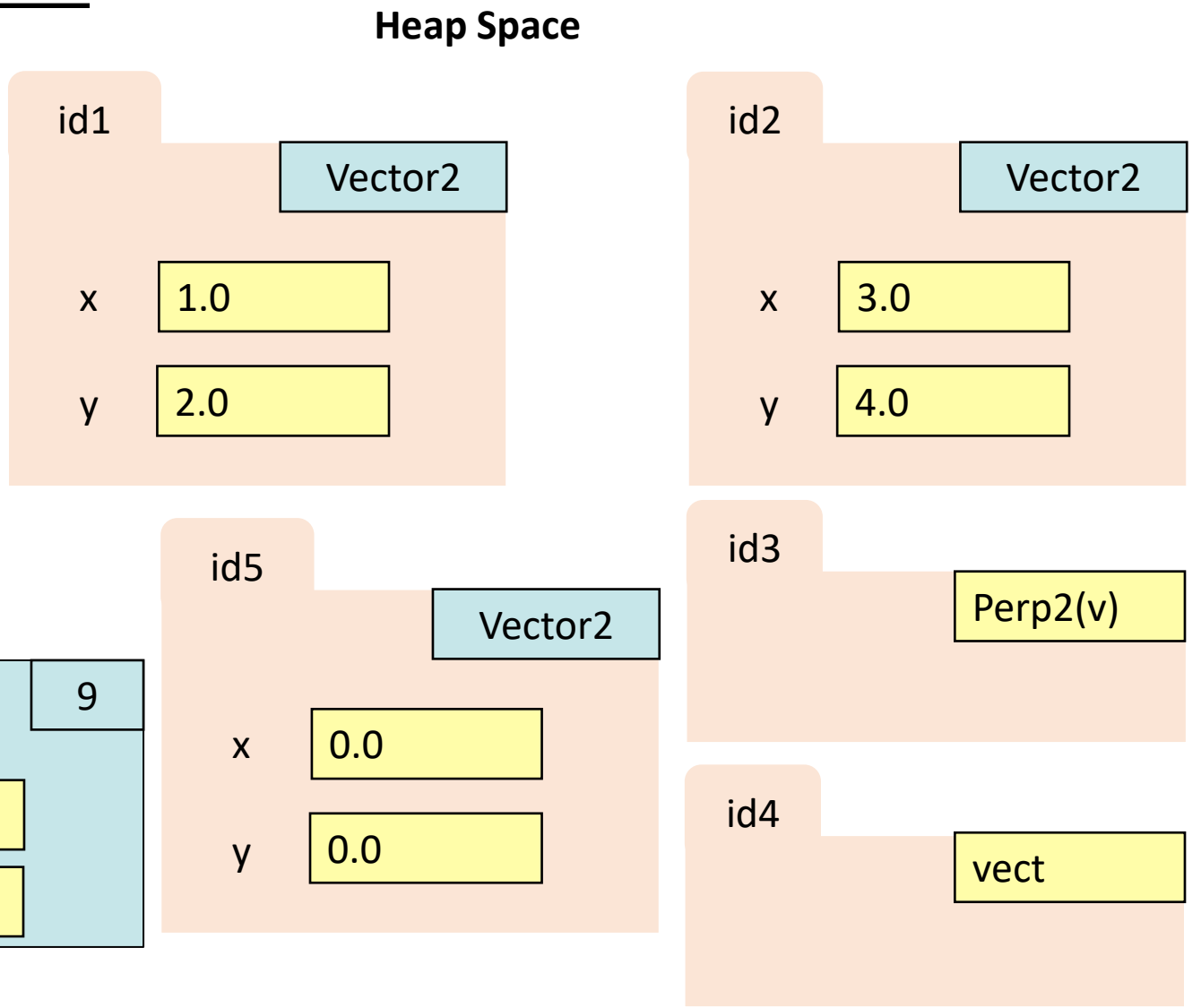


Part D

```
1  Import vect
2  u = Vector2(1.0,2.0)
3  v = Vector2(3.0,4.0)
4  def perp2(v):
5      """Returns a vector perpendicular to v
6
7      Precondition: v is a Vector2 object"""
8      u = vect.Vector2(0.0,0.0)
9      u.y = v.x
10     u.x = -v.y
11     return u
12 v = perp2(u)
```

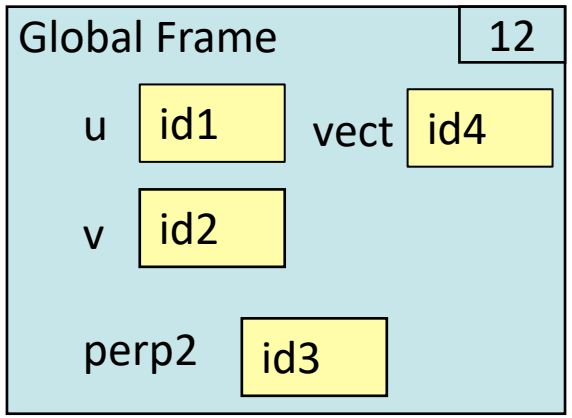


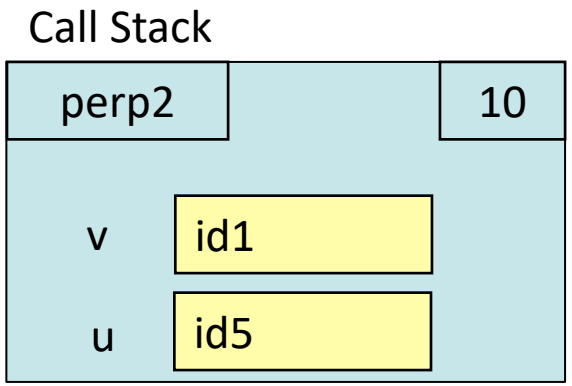


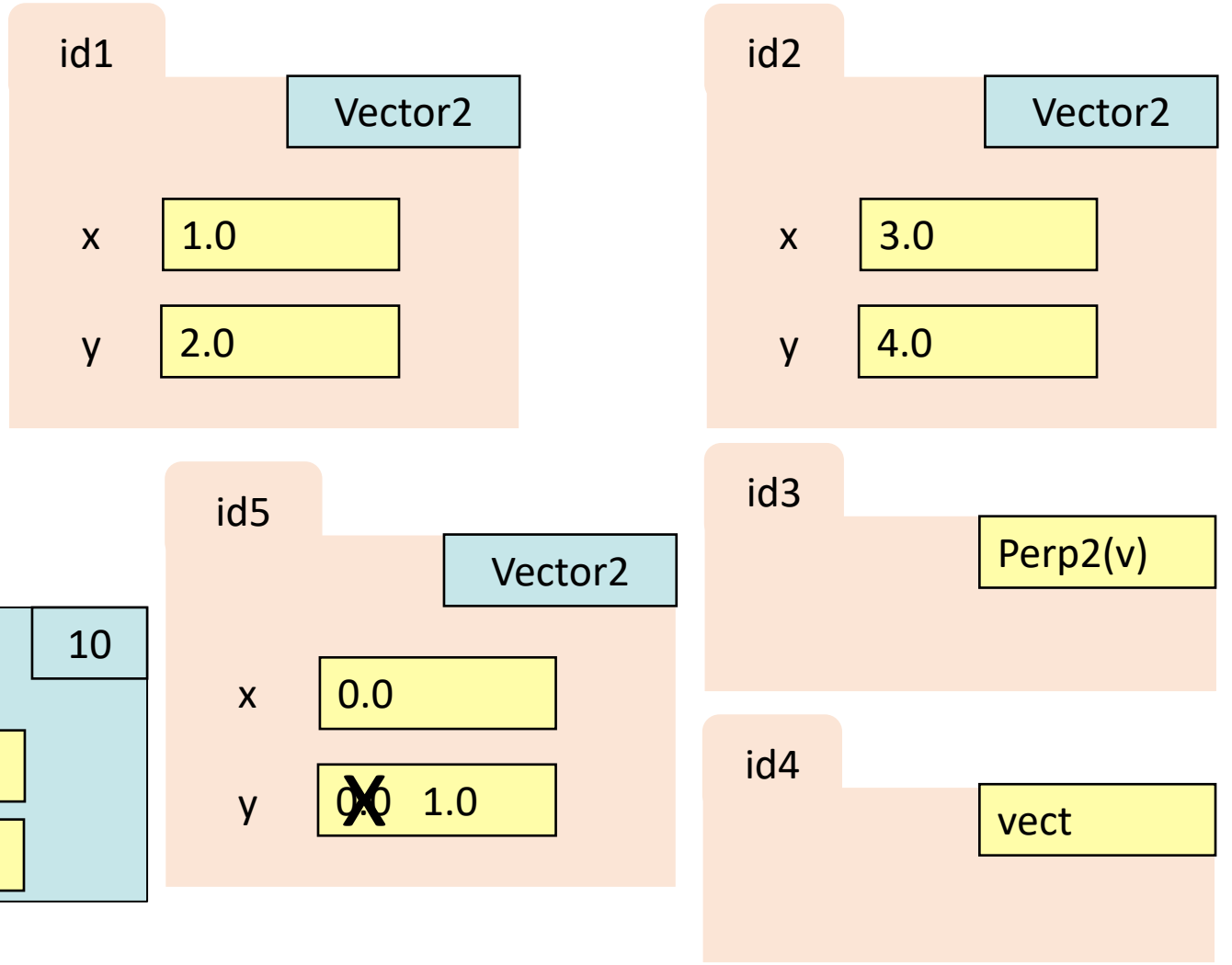


Part D

```
1  Import vect
2  u = Vector2(1.0,2.0)
3  v = Vector2(3.0,4.0)
4  def perp2(v):
5      """Returns a vector perpendicular to v
6
7      Precondition: v is a Vector2 object"""
8      u = vect.Vector2(0.0,0.0)
9      u.y = v.x
10     u.x = -v.y
11     return u
12 v = perp2(u)
```

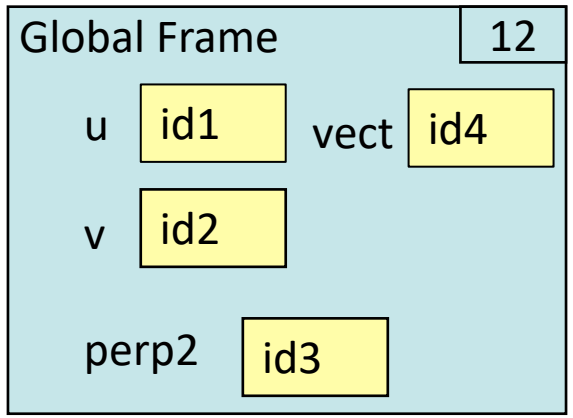


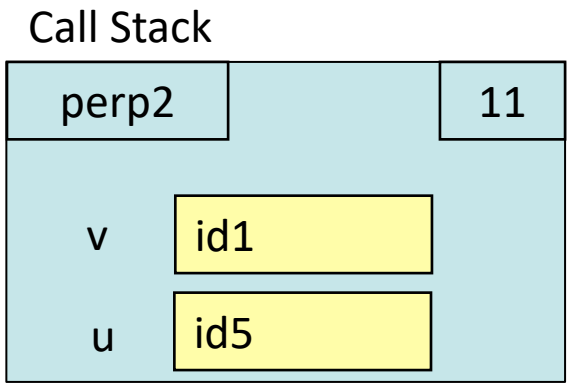


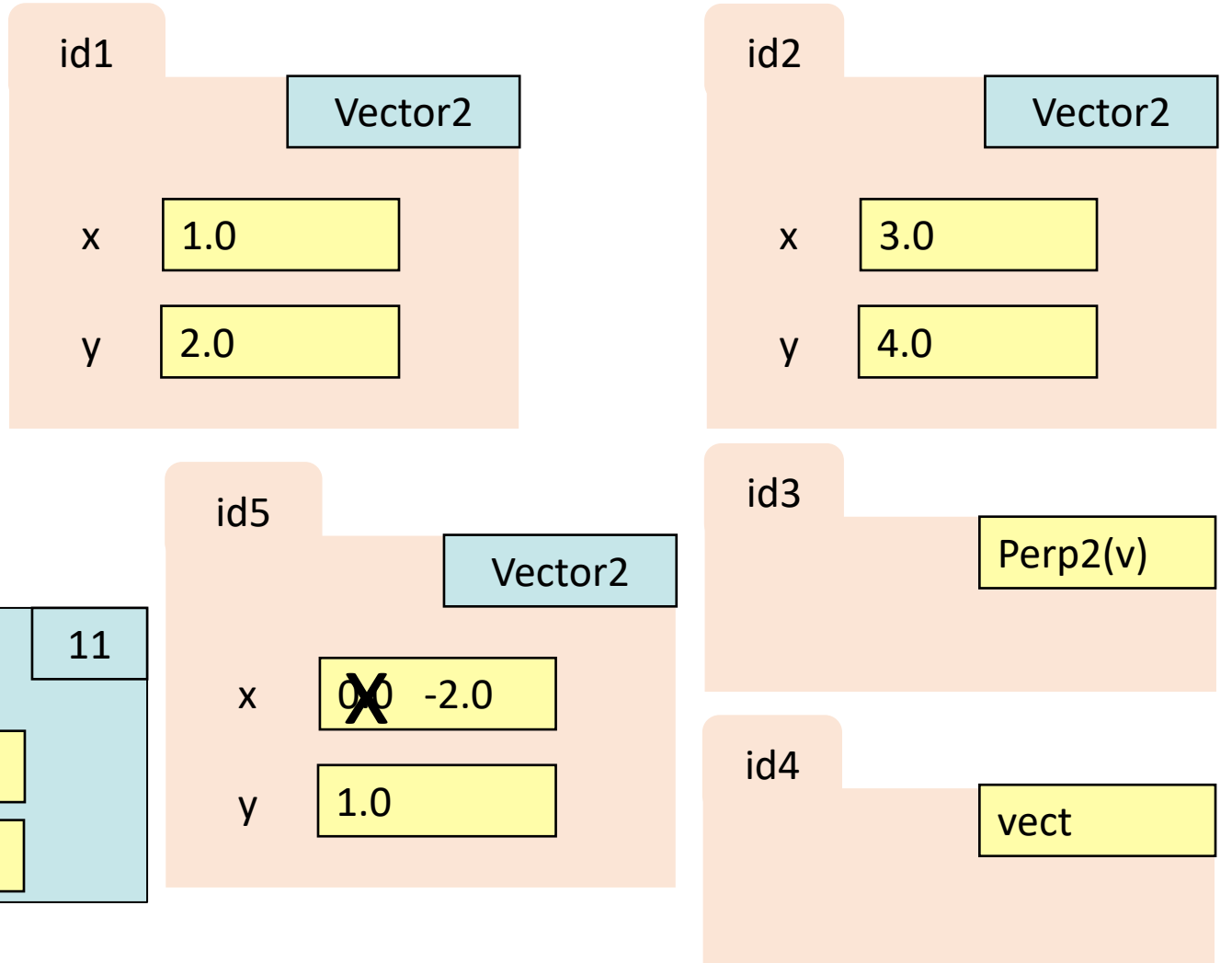


Part D

```
1  Import vect
2  u = Vector2(1.0,2.0)
3  v = Vector2(3.0,4.0)
4  def perp2(v):
5      """Returns a vector perpendicular to v
6
7      Precondition: v is a Vector2 object"""
8      u = vect.Vector2(0.0,0.0)
9      u.y = v.x
10     u.x = -v.y
11     return u
12 v = perp2(u)
```

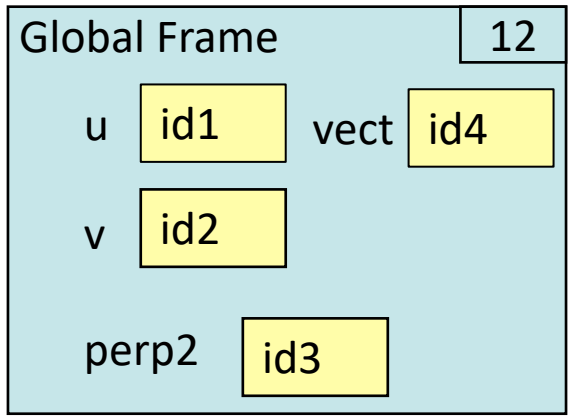


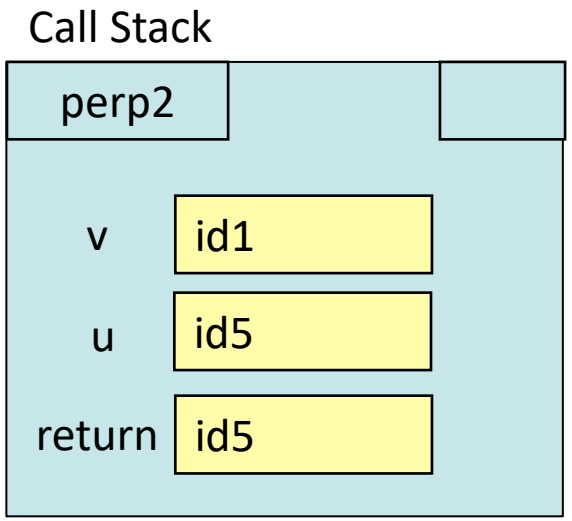


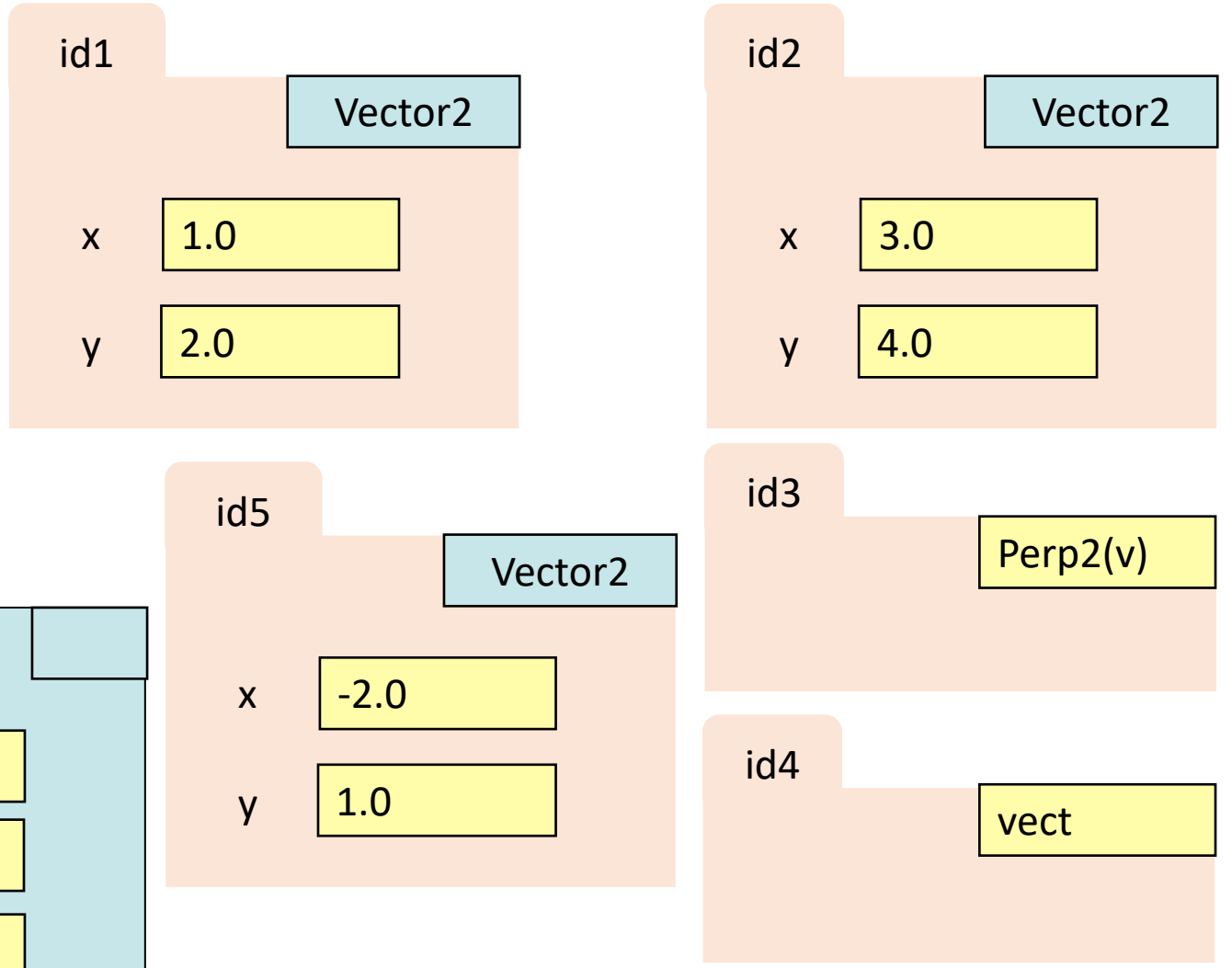


Part D

```
1  Import vect
2  u = Vector2(1.0,2.0)
3  v = Vector2(3.0,4.0)
4  def perp2(v):
5      """Returns a vector perpendicular to v
6
7      Precondition: v is a Vector2 object"""
8      u = vect.Vector2(0.0,0.0)
9      u.y = v.x
10     u.x = -v.y
11     return u
12 v = perp2(u)
```

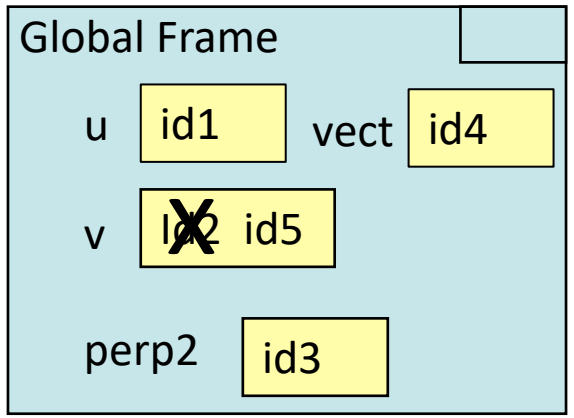


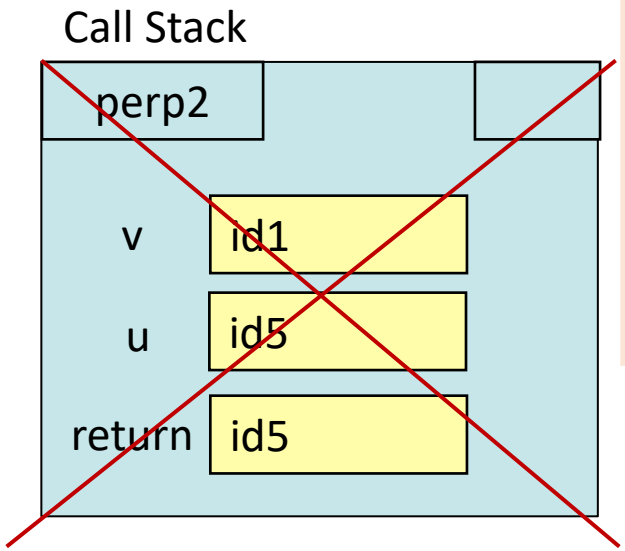


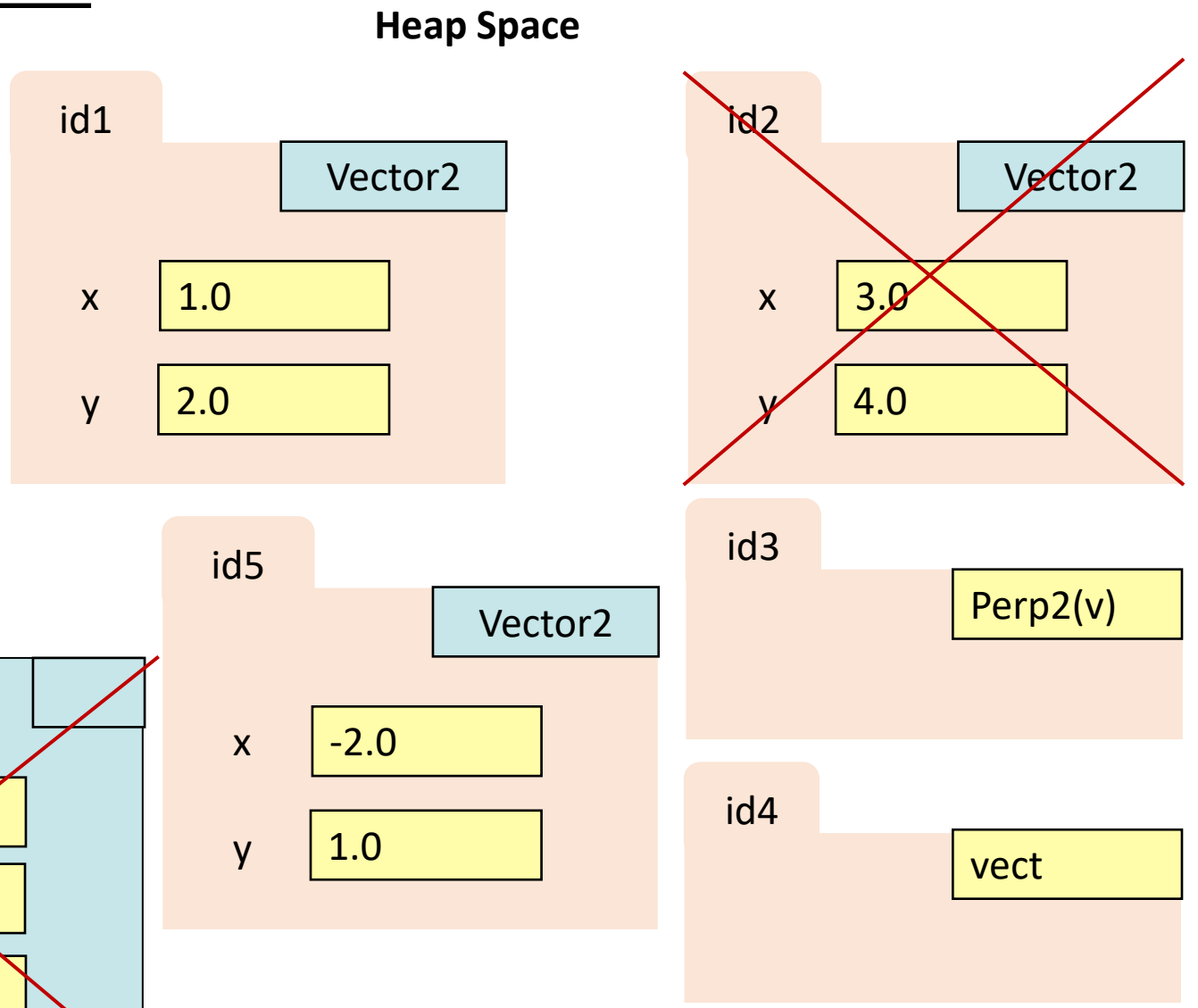


Part D

```
1  Import vect
2  u = Vector2(1.0,2.0)
3  v = Vector2(3.0,4.0)
4  def perp2(v):
5      """Returns a vector perpendicular to v
6
7      Precondition: v is a Vector2 object"""
8      u = vect.Vector2(0.0,0.0)
9      u.y = v.x
10     u.x = -v.y
11     return u
12 v = perp2(u)
```

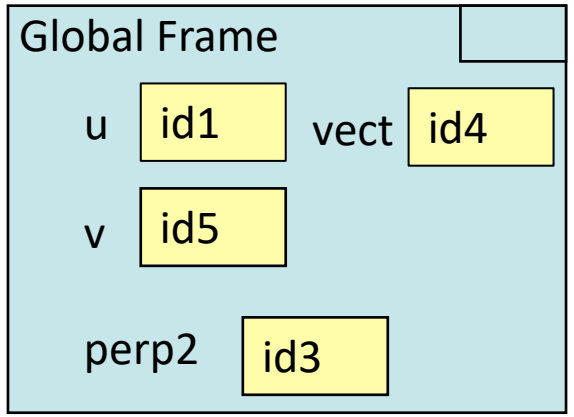


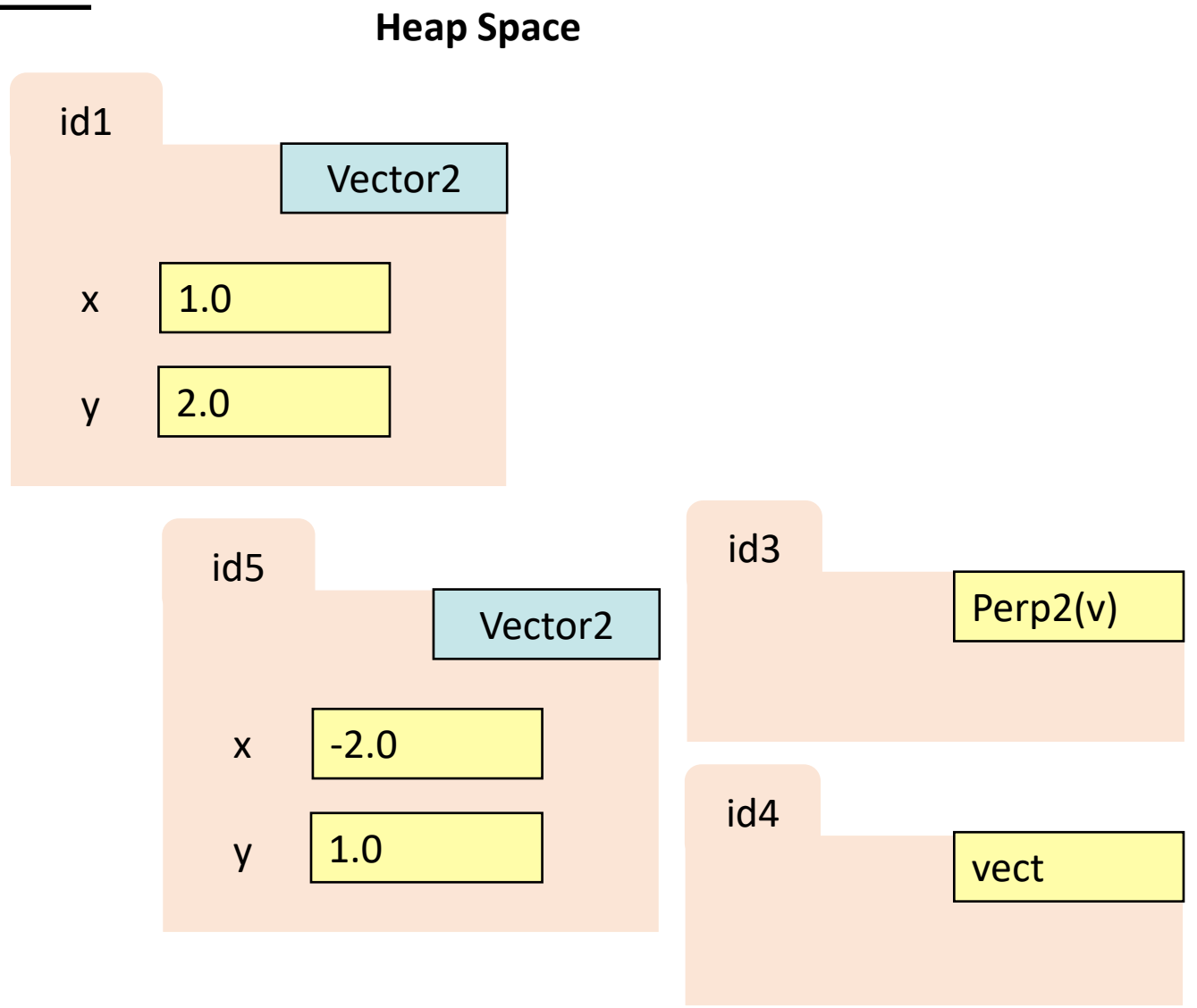




Part D

```
1  Import vect
2  u = Vector2(1.0,2.0)
3  v = Vector2(3.0,4.0)
4  def perp2(v):
5      """Returns a vector perpendicular to v
6
7      Precondition: v is a Vector2 object"""
8      u = vect.Vector2(0.0,0.0)
9      u.y = v.x
10     u.x = -v.y
11     return u
12 v = perp2(u)
```





Part E

```
def dist(x,y):
    """Returns: The number line distance between x and y

    Example: dist(2,5) returns 3
    Example: dist(5,2) returns 3

    Parameter x: the starting point
    Precondition: x is a number

    Parameter y: the ending point
    Precondition: y is a number"""
    a = x-y
    if a < 0:
        b = -a
    else:
        b = a
    return b
dist(3,5)
dist(7,2)
```

Thank You