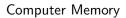
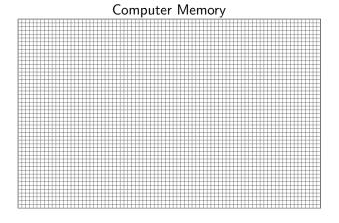
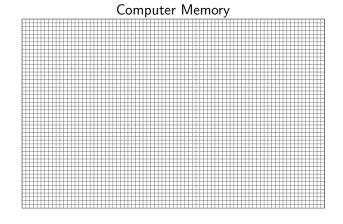
Pointers

Deepanjan Kesh

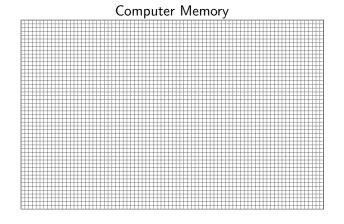
Computer Memory						
	· ·					



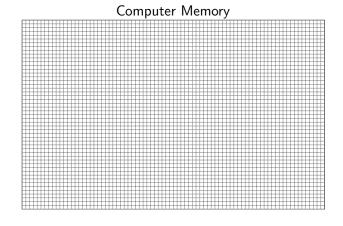




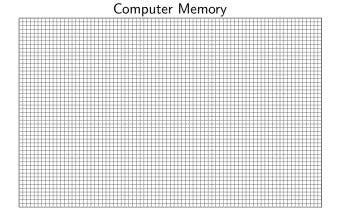
 $256\; \mathsf{GB} = 34359738368\; \mathsf{bytes}$



256 GB = 34359738368 squares



256 GB = 34359738368 unique squares

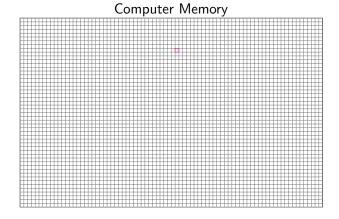


Assign a unique address to each byte

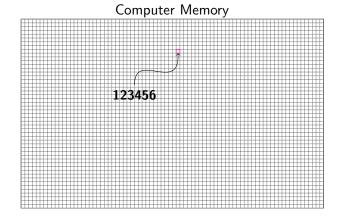
Computer Memory										
I										

Camanutar Managar

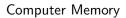
Assign a unique address to each byte Numbers from 0 to 34359738367

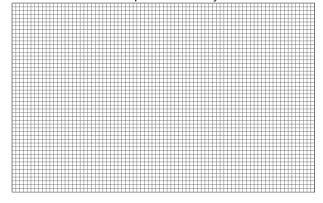


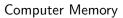
Assign a unique address to each byte Numbers from 0 to 34359738367

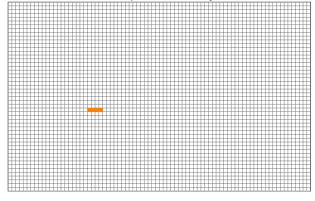


Assign a unique address to each byte Numbers from 0 to 34359738367









Computer Memory								
		ā	a.					

Computer Memory								
			8	ì.				
		123456	123457	123458	123459			

int a; \checkmark

a = 5;

compact. Memory									
			a	ì.					
		123456	123457	123458	123459				

int	a;	✓
a =	5;	/

Compared Memory									
		8	a.						
	123456	123457	123458	123459					

int a; \checkmark

 $a = 5; \checkmark$

				-)	
		ā	a.		
		į			
	123456	123457	123458	123459	

int a; ✓
a = 5; ✓

oompass. memery										
			8	a.						
			10							
		123456	123457	123458	123459					

 $a = 5; \checkmark$

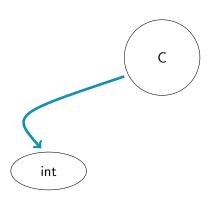
			-)	
	8	ì		
0000000	000000000			
123456	123457	123458	123459	

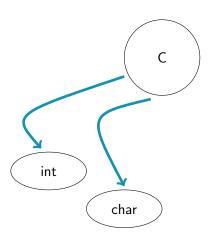
int a; \checkmark

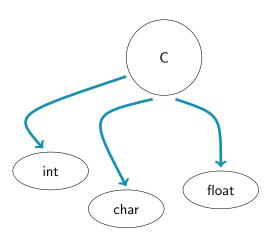
 $a = 5; \checkmark$

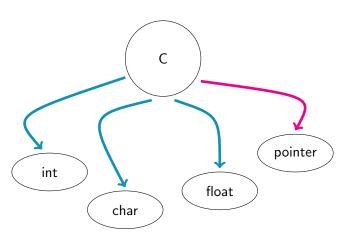
				-)	
		ā	a.		
		į			
	123456	123457	123458	123459	

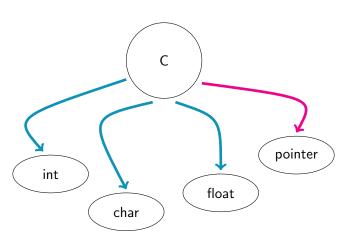




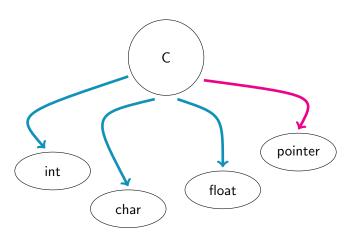








Pointer



Pointer Stores address of variables.

 $a = 5; \checkmark$

		ā	ì							
		Ę								
	123456	123457	123458	123459						

ı	Со	mpute	r Mem	ory	ı	
		8	a.			
		į				
	123456	123457	123458	123459		

int a; ✓
 *p
a = 5; ✓

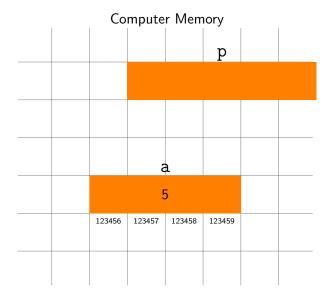
ı	Со	mpute	r Mem	ory	1
		8	a.		
		į			
	123456	123457	123458	123459	

int a; ✓
int *p
a = 5; ✓

int a; ✓
int *p;
a = 5; ✓

Computer Memory								
			ā	a.				
			į					
		123456	123457	123458	123459			

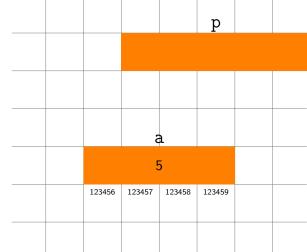
int a; ✓
int *p; ✓
a = 5; ✓



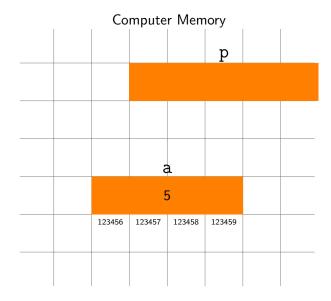
int a;
int *p;
a = 5;
p = a;

Computer Memory

p

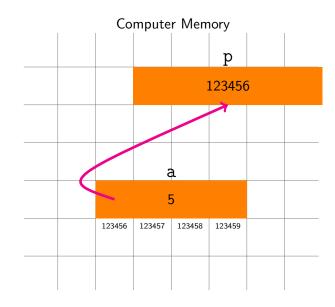


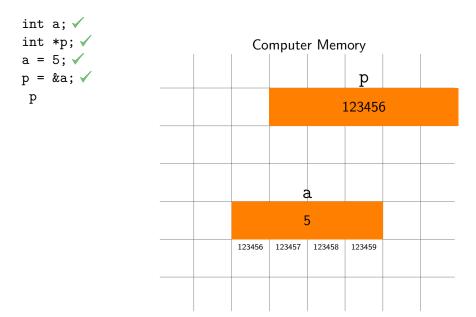
int a; ✓
int *p; ✓
a = 5; ✓
p = a; X



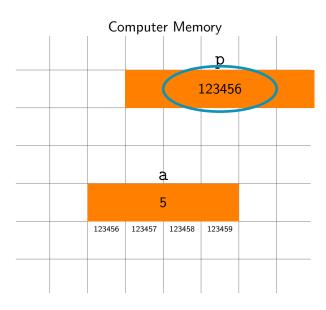
int a; ✓ int *p; ✓ Computer Memory $a = 5; \checkmark$ p = &a;p 123456 123457 123458 123459 int a; ✓ int *p; ✓ Computer Memory $a = 5; \checkmark$ p = &a;p 123456 123457 123458 123459 int a; ✓ int *p; ✓ Computer Memory $a = 5; \checkmark$ p = &a; ✓ p 123456 а 123456 123457 123458 123459

int a; int *p; a = 5; p = &a;

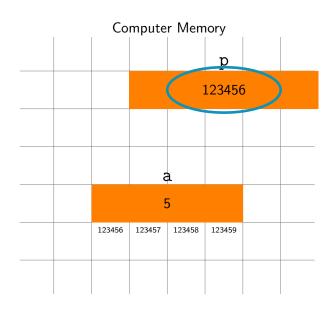




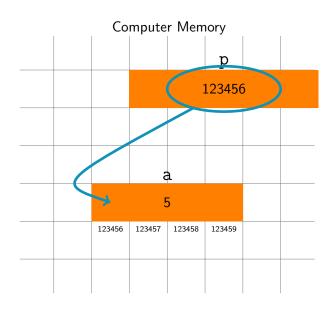
int a;
int *p;
a = 5;
p = &a;
p



int a; \/
int *p; \/
a = 5; \/
p = &a; \/
*p

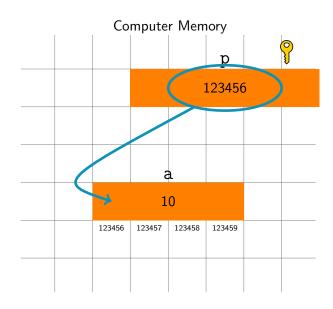


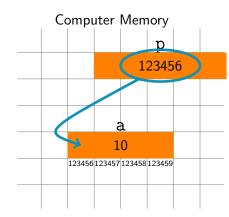
int a;
int *p;
a = 5;
p = &a;
*p



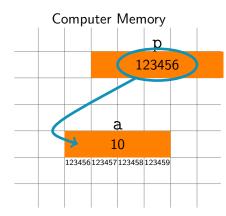
int a; ✓ int *p; ✓ Computer Memory $a = 5; \checkmark$ p = &a; ✓ *p = 10;123456 123456 123457 123458 123459

```
int a; ✓
int *p; ✓
a = 5; ✓
p = &a; ✓
*p = 10; ✓
```

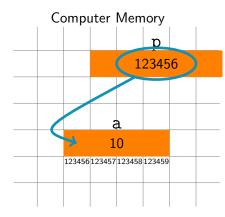




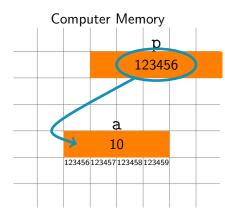
• Every variable has some value.



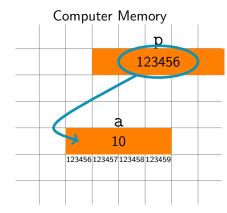
- Every variable has some value.
 - printf("%d", a)



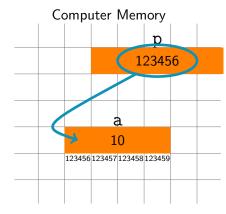
- Every variable has some value.
 - \bullet printf("%d", a) \rightarrow 10



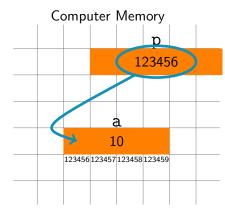
- Every variable has some value.
 - ullet printf("%d", a) ightarrow 10
 - printf(" ", p)



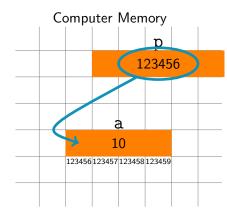
- Every variable has some value.
 - ullet printf("%d", a) ightarrow 10
 - printf("%p", p)



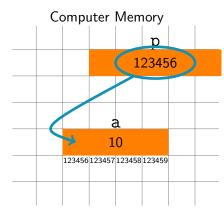
- Every variable has some value.
 - ullet printf("%d", a) ightarrow 10
 - printf("%p", p) ightarrow 123456



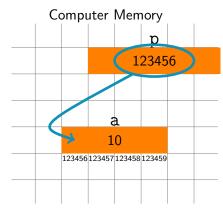
- Every variable has some value.
 - ullet printf("%d", a) ightarrow 10
 - printf("%p", p) ightarrow 123456
- Every variable has an address.



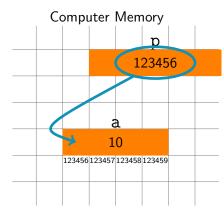
- Every variable has some value.
 - printf("%d", a) \rightarrow 10
 - printf("%p", p) ightarrow 123456
- Every variable has an address.
 - Address of a is 123456 to 123459.



- Every variable has some value.
 - printf("%d", a) \rightarrow 10
 - printf("%p", p) ightarrow 123456
- Every variable has an address.
 - Address of a is 123456 to 123459.
 - printf(" ",)

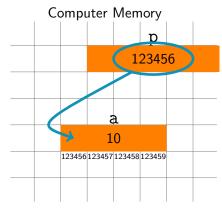


- Every variable has some value.
 - printf("%d", a) \rightarrow 10
 - printf("%p", p) ightarrow 123456
- Every variable has an address.
 - Address of a is 123456 to 123459.
 - printf(" ", &a)



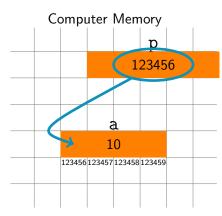
- Every variable has some value.
 - printf("%d", a) \rightarrow 10
 - printf("%p", p) ightarrow 123456
- Every variable has an address.
 - Address of a is 123456 to 123459.

printf("%p", &a)

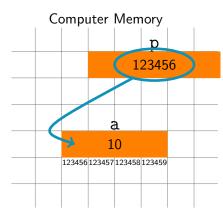


- Every variable has some value.
 - printf("%d", a) \rightarrow 10
 - printf("%p", p) \rightarrow 123456
- Every variable has an address.
 - Address of a is 123456 to 123459.

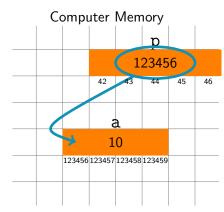
printf("%p", &a) \rightarrow 123456



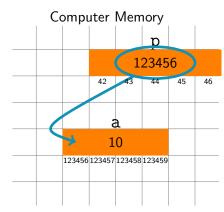
- Every variable has some value.
 - printf("%d", a) \rightarrow 10
 - printf("%p", p) ightarrow 123456
- Every variable has an address.
 - Address of a is 123456 to 123459.
 printf("%p", &a) →
 - printf("%p", &a) \rightarrow 123456
 - Address of p?



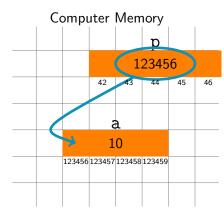
- Every variable has some value.
 - printf("%d", a) \rightarrow 10
 - printf("%p", p) ightarrow 123456
- Every variable has an address.
 - Address of a is 123456 to 123459.
 printf("%p", &a) → 123456
 - Address of p?



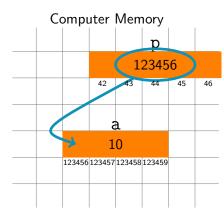
- Every variable has some value.
 - printf("%d", a) \rightarrow 10
 - printf("%p", p) \rightarrow 123456
- Every variable has an address.
 - Address of a is 123456 to 123459.
 printf("%p", &a) → 123456
 - Address of p is 42 to 46.



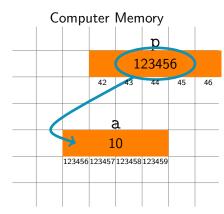
- Every variable has some value.
 - printf("%d", a) \rightarrow 10
 - printf("%p", p) \rightarrow 123456
- Every variable has an address.
 - Address of a is 123456 to 123459.
 printf("%p", &a) → 123456
 - Address of p is 42 to 46. printf(" ",)



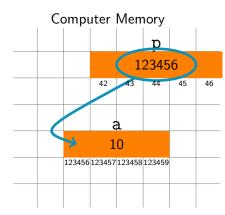
- Every variable has some value.
 - printf("%d", a) \rightarrow 10
 - printf("%p", p) \rightarrow 123456
- Every variable has an address.
 - Address of a is 123456 to 123459.
 printf("%p", &a) → 123456
 - Address of p is 42 to 46. printf(" ", &p)



- Every variable has some value.
 - printf("%d", a) \rightarrow 10
 - printf("%p", p) \rightarrow 123456
- Every variable has an address.
 - Address of a is 123456 to 123459.
 printf("%p", &a) → 123456
 - Address of p is 42 to 46. printf("%p", &p)



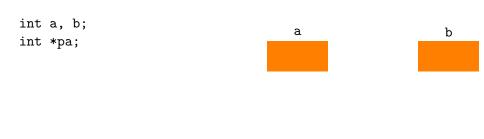
- Every variable has some value.
 - printf("%d", a) \rightarrow 10
 - printf("%p", p) \rightarrow 123456
- Every variable has an address.
 - Address of a is 123456 to 123459. printf("%p", &a) \rightarrow 123456
 - Address of p is 42 to 46.
 - printf("%p", &p) \rightarrow 42



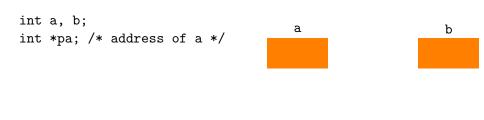
int a, b;

int a, b;

a b



int a, b;
int *pa; /* pointer to a */
b



int a, b;
int *pa; /* address of a */



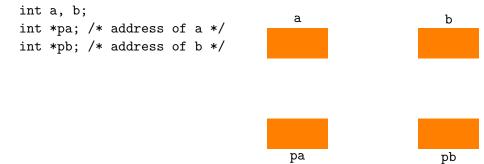
int a, b;
int *pa; /* address of a */
int *pb;

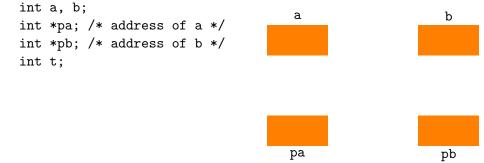
a
b

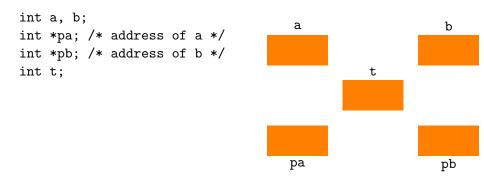


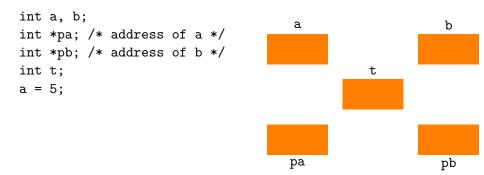
int a, b;
int *pa; /* address of a */
int *pb; /* address of b */

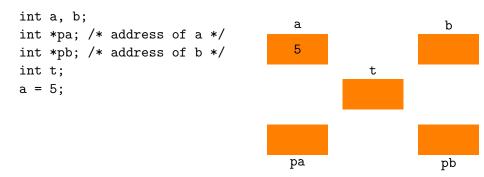


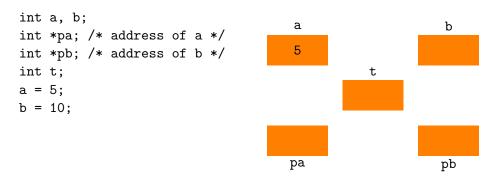


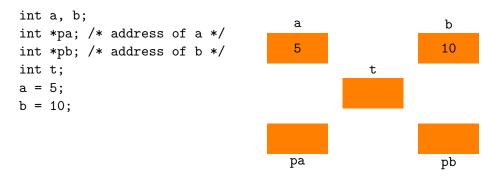


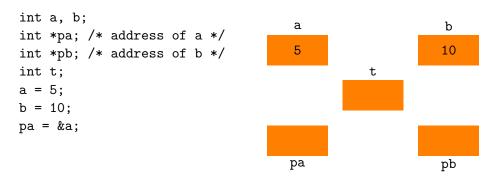


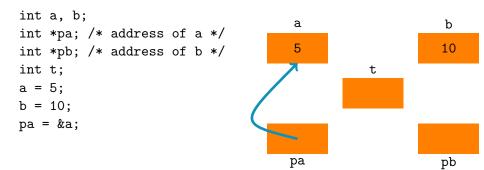


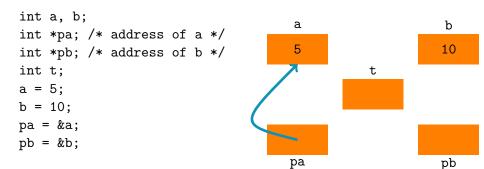


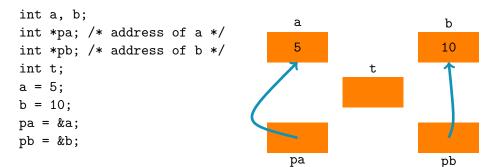












```
int a, b;
int *pa; /* address of a */
int *pb; /* address of b */
int t;
a = 5;
b = 10;
pa = &a;
pb = &b;

pa

b

10

10

pa

pb
```

pa

```
int a, b;
int *pa; /* address of a */
int *pb; /* address of b */
int t;
a = 5;
b = 10;
pa = &a;
pb = &b;

pa

b

10

t
pa

pb
```

pa

```
int a, b;
int *pa; /* address of a */
int *pb; /* address of b */
int t;
a = 5;
b = 10;
pa = &a;
pb = &b;

b

10

10

pa

pb
```

*pa

```
int a, b;
int *pa; /* address of a */
int *pb; /* address of b */
int t;
a = 5;
b = 10;
pa = &a;
pb = &b;

pa pb
```

*pa

```
int a, b;
int *pa; /* address of a */
int *pb; /* address of b */
int t;
a = 5;
b = 10;
pa = &a;
pb = &b;

pa pb
```

t = *pa;

```
int a, b;
int *pa; /* address of a */
int *pb; /* address of b */
int t;
a = 5;
b = 10;
pa = &a;
pb = &b;

pa

b

10

t
pa
pb
```

t = *pa;

```
int a, b;
int *pa; /* address of a */
int *pb; /* address of b */
int t;
a = 5;
b = 10;
pa = &a;
pb = &b;

pa

pb
```

t = *pa;

```
int a, b;
int *pa; /* address of a */
int *pb; /* address of b */
int t;
a = 5;
b = 10;
pa = &a;
pb = &b;

pa

b

10

10

pa

pb
```

```
t = *pa;
    pb
```

```
int a, b;
int *pa; /* address of a */
int *pb; /* address of b */
int t;
a = 5;
b = 10;
pa = &a;
pb = &b;

pa
```

```
t = *pa;
    pb
```

```
int a, b;
int *pa; /* address of a */
int *pb; /* address of b */
int t;
a = 5;
b = 10;
pa = &a;
pb = &b;

a

b

10

10
```

```
t = *pa;
*pb
```

```
int a, b;
int *pa; /* address of a */
int *pb; /* address of b */
int t;
a = 5;
b = 10;
pa = &a;
pb = &b;
pa

pb
```

```
t = *pa;
*pb
```

```
int a, b;
int *pa; /* address of a */
int *pb; /* address of b */
int t;
a = 5;
b = 10;
pa = &a;
pb = &b;

pa

pb
```

```
t = *pa;
*pa = *pb;
```

```
int a, b;
int *pa; /* address of a */
int *pb; /* address of b */
int t;
a = 5;
b = 10;
pa = &a;
pb = &b;

pa

b

10

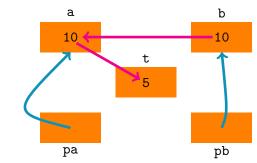
10

pa

pb
```

```
t = *pa;
*pa = *pb;
```

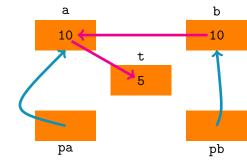
```
int a, b;
int *pa; /* address of a */
int *pb; /* address of b */
int t;
a = 5;
b = 10;
pa = &a;
```



```
t = *pa;
*pa = *pb;
```

pb = &b;

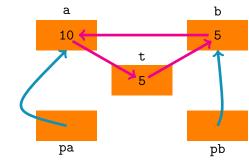
```
int a, b;
int *pa; /* address of a */
int *pb; /* address of b */
int t;
a = 5;
b = 10;
pa = &a;
pb = &b;
```



```
t = *pa;
*pa = *pb;
```

*pb = t;

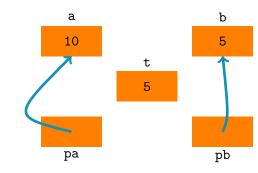
```
int a, b;
int *pa; /* address of a */
int *pb; /* address of b */
int t;
a = 5;
b = 10;
pa = &a;
```



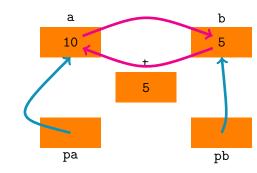
```
t = *pa;
*pa = *pb;
```

pb = &b;

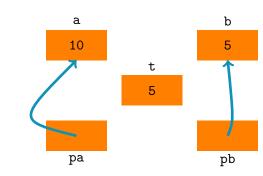
*pb = t;



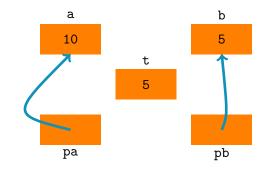
```
t = *pa;
*pa = *pb;
*pb = t;
```



```
t = *pa;
*pa = *pb;
*pb = t;
```

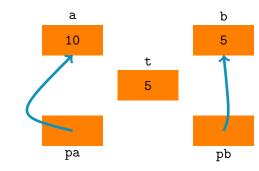


```
swap(
{
  t = *pa;
  *pa = *pb;
  *pb = t;
}
```

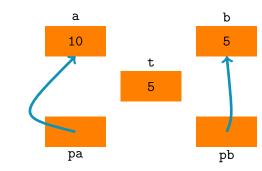


```
swap(int *pa, int *pb)

t = *pa;
*pa = *pb;
*pb = t;
```



```
void swap(int *pa, int *pb)
{
   t = *pa;
   *pa = *pb;
   *pb = t;
}
```



```
void swap(int *pa, int *pb)
{
  int t;
  t = *pa;
  *pa = *pb;
  *pb = t;
}
```

```
int a, b;
                                                                 b
                                          a
  int *pa;
                                          10
                                                                 5
  int *pb;
  int t;
                                                     t
  a = 5;
                                                     5
  b = 10;
  pa = &a;
  pb = \&b;
                                          pa
                                                                pb
void swap(int *pa, int *pb)
  int t;
  t = *pa;
  *pa = *pb;
  *pb = t;
```

```
int a, b;
                                                                b
                                          a
  int *pa;
                                          10
  int *pb;
  int t;
                                                     t
  a = 5;
                                                     5
  b = 10;
  pa = &a;
  pb = \&b;
                                         pa
                                                               pb
void swap(int *pa, int *pb)
  int t;
  t = *pa;
  *pa = *pb;
  *pb = t;
```

```
int a, b;
                                                                b
                                          a
  int *pa;
                                         10
  int *pb;
  int t;
                                                     t
  a = 5;
                                                     5
  b = 10;
  pa = &a;
  pb = \&b;
                                         pa
                                                               pb
void swap(int *pa, int *pb)
  int t;
  t = *pa;
  *pa = *pb;
  *pb = t;
```

```
int a, b;
                                                                b
                                          a
  int *pa;
                                         10
  int *pb;
  int t;
                                                     t
  a = 5;
                                                     5
  b = 10;
  pa = &a;
  pb = \&b;
                                         pa
                                                               pb
void swap(int *pa, int *pb)
  int t;
  t = *pa;
  *pa = *pb;
  *pb = t;
```

```
int a, b;
                                                                    b
                                             a
  int *pa;
                                            10
  int *pb;
  int t;
                                                        t
  a = 5;
                                                        5
  b = 10;
  <del>pa = &a;</del>
  pb = \&b;
                                            pa
                                                                    pb
void swap(int *pa, int *pb)
  int t;
  t = *pa;
  *pa = *pb;
  *pb = t;
```

```
int a, b;
                                                                    b
                                             a
  int *pa;
                                            10
  int *pb;
  int t;
                                                        t
  a = 5;
                                                        5
  b = 10;
  <del>pa = &a;</del>
  pb = \&b;
                                            pa
                                                                   pb
void swap(int *pa, int *pb)
  int t;
  t = *pa;
  *pa = *pb;
  *pb = t;
```

```
int a, b;
                                            a
                                                                    b
  int *pa;
                                            10
  int *pb;
  int t;
                                                        t
  a = 5;
                                                        5
  b = 10;
  <del>pa = &a;</del>
  pb = \&b;
  swap(&a, &b);
                                            pa
                                                                   pb
void swap(int *pa, int *pb)
  int t;
  t = *pa;
  *pa = *pb;
  *pb = t;
```

```
int main(void)
  int a, b;
                                         а
                                                               b
  int *pa;
                                         10
                                                               5
  int *pb;
  int t;
                                                    t
  a = 5;
                                                    5
  b = 10;
  pa = &a;
  pb = \&b;
  swap(&a, &b);
                                         рa
                                                               pb
  return 0;
void swap(int *pa, int *pb)
  int t;
  t = *pa;
  *pa = *pb;
  *pb = t;
```

/* area and perimeter of a rectange */

```
/* area and perimeter of a rectange */
         area_peri(
{
```

```
/* area and perimeter of a rectange */
    area_peri(int len
{
```

```
/* area and perimeter of a rectange */
    area_peri(int len, int bth
{
```

```
/* area and perimeter of a rectange */
    area_peri(int len, int bth, int *ar
{
```

```
/* area and perimeter of a rectange */
    area_peri(int len, int bth, int *ar, int *pr)
{
```

```
/* area and perimeter of a rectange */
    area_peri(int len, int bth, int *ar, int *pr)
{
    *ar = len * bth;
}
```

```
/* area and perimeter of a rectange */
    area_peri(int len, int bth, int *ar, int *pr)
{
    *ar = len * bth;
    *pr = 2 * (len + bth);
```

```
/* area and perimeter of a rectange */
void area_peri(int len, int bth, int *ar, int *pr)
{
   *ar = len * bth;
   *pr = 2 * (len + bth);
```

```
/* area and perimeter of a rectange */
void area_peri(int len, int bth, int *ar, int *pr)
{
   *ar = len * bth;
   *pr = 2 * (len + bth);
}
int main(void)
{
```

```
/* area and perimeter of a rectange */
void area_peri(int len, int bth, int *ar, int *pr)
{
   *ar = len * bth;
   *pr = 2 * (len + bth);
}
int main(void)
{
   int l, b;
```

```
/* area and perimeter of a rectange */
void area_peri(int len, int bth, int *ar, int *pr)
  *ar = len * bth;
  *pr = 2 * (len + bth);
int main(void)
  int 1, b;
  int a, p;
```

```
/* area and perimeter of a rectange */
void area_peri(int len, int bth, int *ar, int *pr)
  *ar = len * bth;
  *pr = 2 * (len + bth);
int main(void)
  int 1, b;
  int a, p;
  1 = 2;
```

```
/* area and perimeter of a rectange */
void area_peri(int len, int bth, int *ar, int *pr)
  *ar = len * bth;
  *pr = 2 * (len + bth);
int main(void)
  int 1, b;
  int a, p;
  1 = 2;
  b = 3;
```

```
/* area and perimeter of a rectange */
void area_peri(int len, int bth, int *ar, int *pr)
  *ar = len * bth;
  *pr = 2 * (len + bth);
int main(void)
  int 1, b;
  int a, p;
  1 = 2;
  b = 3;
  area_peri(1, b, &a, &p);
```

```
/* area and perimeter of a rectange */
void area_peri(int len, int bth, int *ar, int *pr)
  *ar = len * bth;
  *pr = 2 * (len + bth);
int main(void)
  int 1, b;
  int a, p;
  1 = 2;
  b = 3:
  area_peri(1, b, &a, &p);
  printf("area = %d, perimeter = %d\n", a, p);
```

```
#include <stdio.h>
/* area and perimeter of a rectange */
void area_peri(int len, int bth, int *ar, int *pr)
  *ar = len * bth;
  *pr = 2 * (len + bth);
int main(void)
  int 1, b;
  int a, p;
  1 = 2;
  b = 3:
  area_peri(1, b, &a, &p);
  printf("area = %d, perimeter = %d\n", a, p);
```

```
#include <stdio.h>
/* area and perimeter of a rectange */
void area_peri(int len, int bth, int *ar, int *pr)
  *ar = len * bth;
  *pr = 2 * (len + bth);
int main(void)
  int 1, b;
  int a, p;
  1 = 2;
  b = 3:
  area_peri(1, b, &a, &p);
  printf("area = %d, perimeter = %d\n", a, p);
  return 0;
```