COMP2012H
Honors Object-Oriented Programming & Data Structures

Suggested Solution of Practice Problems
~ Pointers and Memory Allocation

Prepared by Desmond Yau-Chat TSOI
Lecturer, Department of Computer Science and Engineering,
The Hong Kong University of Science and Technology

Lecture notes are based on textbook and various resources over the Internet

Question 1

- State the problem(s) for the code.
  Also, suggest how to fix the problems.

```
#include <iostream>
using namespace std;
int main( ) {
   int i=1, j=2, k=3, n=4, m=5;
   int *x, *y, *z;
   *x = &i;
   cout << "x: " << x << endl;
   cout << "*x: " << *x << endl;
   cout << "&x: " << &x << endl;
   cout << "i: " << i << endl;
   cout << "*i: " << *i << endl;
   cout << "&i: " << &i << endl;
   return 0;
}
```

x = &i;

Question 2

- State the problem(s) for the code.
  Also, suggest how to fix the problems.

```
#include <iostream>
using namespace std;
int main() {
   int x = 2;
   int* px;
   *px = &x;
   cout << *px << endl;
   return 0;
}
```

px = &x;

Question 3

- State the problem(s) for the code.
  Also, suggest how to fix the problems.

```
#include <iostream>
using namespace std;
int main() {
   int x = 2, y = 10;
   int* px, py;
   px = &x;
   py = &y;
   cout << *px << endl;
   cout << *py << endl;
   return 0;
}
```

*py;

OR

```
#include <iostream>
using namespace std;
int main() {
   int x = 2, y = 10;
   int* px, py;
   px = &x;
   py = &y;
   cout << *px << endl;
   cout << *py << endl;
   return 0;
}
```

py = y;

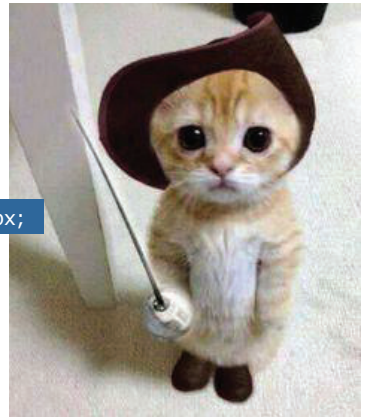
cout << py << endl;

Question 4

- State the problem(s) for the code. Also, suggest how to fix the problems.

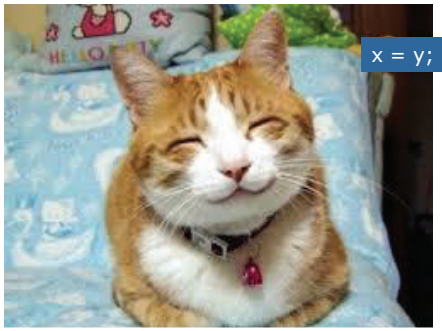
```
#include <iostream>
using namespace std;
int main() {
   int x = 10;
   int* px = &x;
   int* py = &px;
   cout << "x: " << x << endl;
   cout << "*px: " << *px << endl;
   cout << "*py: " << *py << endl;
   return 0;
}
```

int* py = px;



Question 5

- State the problem(s) for the code. Also, suggest how to fix the problems.



x = y;

```
#include <iostream>
using namespace std;
int main() {
   int i=1, j=5, k=7, n=9, m=18;
   int *x, *y, *z;
   x = &i;
   y = z = &j;
   z = x;
   *z = 8;
   y = &k;
   x = &y;
   n = 5;
   *x = 4;
   cout << "i: " << i << endl;
   cout << "j: " << j << endl;
   cout << "k: " << k << endl;
   cout << "n: " << n << endl;
   cout << "m: " << m << endl;
   cout << "*x: " << *x << endl;
   cout << "*y: " << *y << endl;
   cout << "*z: " << *z << endl;
   return 0;
}
```

Question 6

- State the problem(s) for the code. Also, suggest how to fix the problems.

Content of ptr1 cannot be changed, since ptr1 is a constant pointer

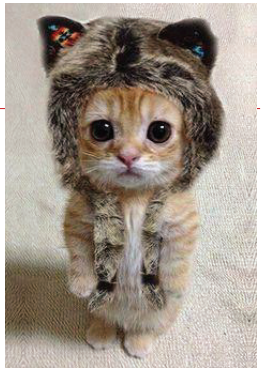
ptr2 stores the address of variable d, and the content of d cannot be changed via ptr2

```
#include <iostream>
using namespace std;
int main() {
   int i=1, j=2, k=3, n=4, m=5;
   int *x, *y, *z;
   x = &i;
   y = z = &j;
   z = x;
   char c = 'c';
   const char d = 'd';
   char * const ptr1 = &c;
   ptr1 = &d;
   const char *ptr2 = &d;
   *ptr2 = 'e';
   return 0;
}
```


Question 7

- Consider two pointers xp1 and xp2, which are defined as follows:

```
int x = 20, y = 30;
const int* xp1 = &x;
int* const xp2 = &y;
```



Which of the following statements is / are **NOT** valid?

a) xp1 = &y; ✓
b) xp2 = xp1; ✗ — As xp2 is a constant pointer variable, it cannot be assigned with other address
c) *xp1 += *xp2; ✗ — xp1 has been constrained, in which it is not able to be used to change the value of the variable it points to, i.e. x. So, *xp1 += *xp2; is wrong
d) *xp2 += *xp1; ✓

b & c are NOT valid

Question 8

- Consider the following code:

```
int main() {
   int x = 20;
   int& y = x;
   int* px = &x;
   int** ppx = &px;
   int*& py = px;
   x++;
   return 0;
}
```

a) Describe each of the variables y, px, ppx, and py in relation to variable x.

- y is a **reference variable**, which is an alias of variable x
- Variable px is a **pointer variable**, which stores the address of variable x
- Variable ppx is a **pointer variable**, which stores the address of variable px, which points to the variable x
- py is a **reference variable**, which is an alias of variable px, which points to the variable x

Question 8 (Cont'd)

- Consider the following code:

```
int main() {
   int x = 20;
   int& y = x;
   int* px = &x;
   int** ppx = &px;
   int*& py = px;
   x++;
   return 0;
}
```

b) Use each of the variables y, px, ppx, and py to write an equivalent statement to x++;

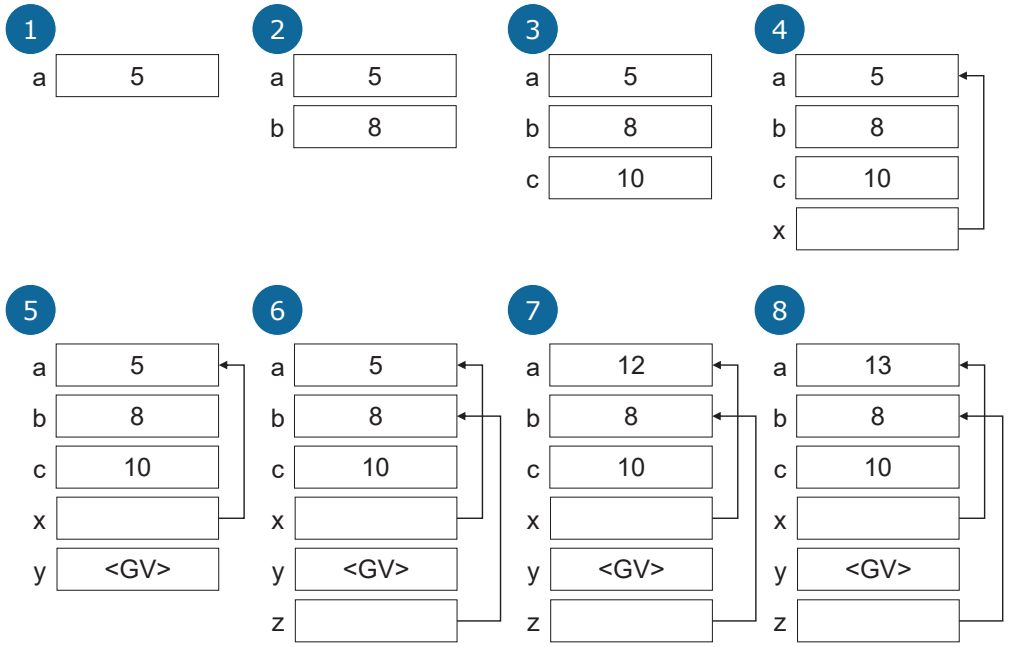
- y++;
- (*px)++;
- (**ppx)++;
- (*py)++;

Question 9

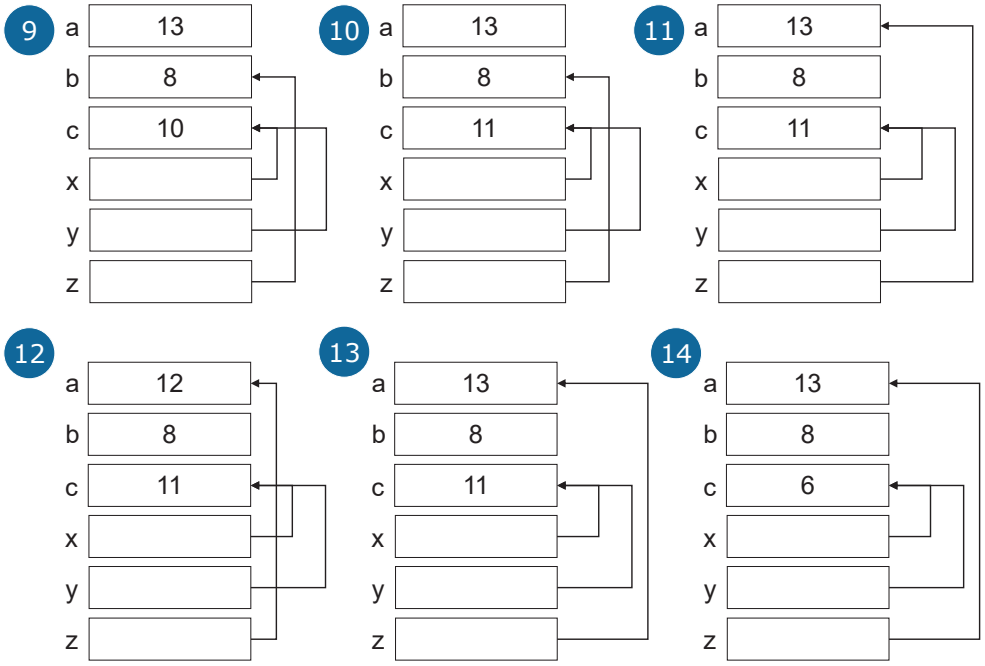
- What is the output of the following program?

```
#include <iostream>
using namespace std;
int main() {
   int a = 5, b = 8, c = 10;
   int* x = &a;
   int* y;
   int* z = &b;
   *x = 12;
   ++(*x);
   y = x = &c;
   c++;
   z = &a;
   --a;
   (*z)++;
   *y = 6;
   cout << a << " " << b << " " << c << endl;
   cout << *x << " " << *y << " " << *z << endl;
   return 0;
}
```

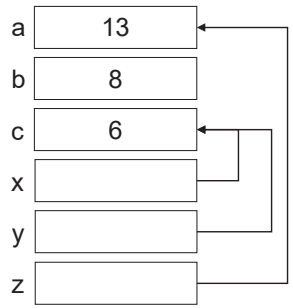
Question 9



Question 9



Question 9



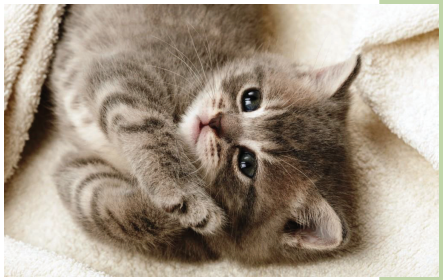
15

```
cout << a << " " << b << " " << c << endl;
cout << *x << " " << *y << " " << *z << endl;
```

```
13 8 6
6 6 13
```

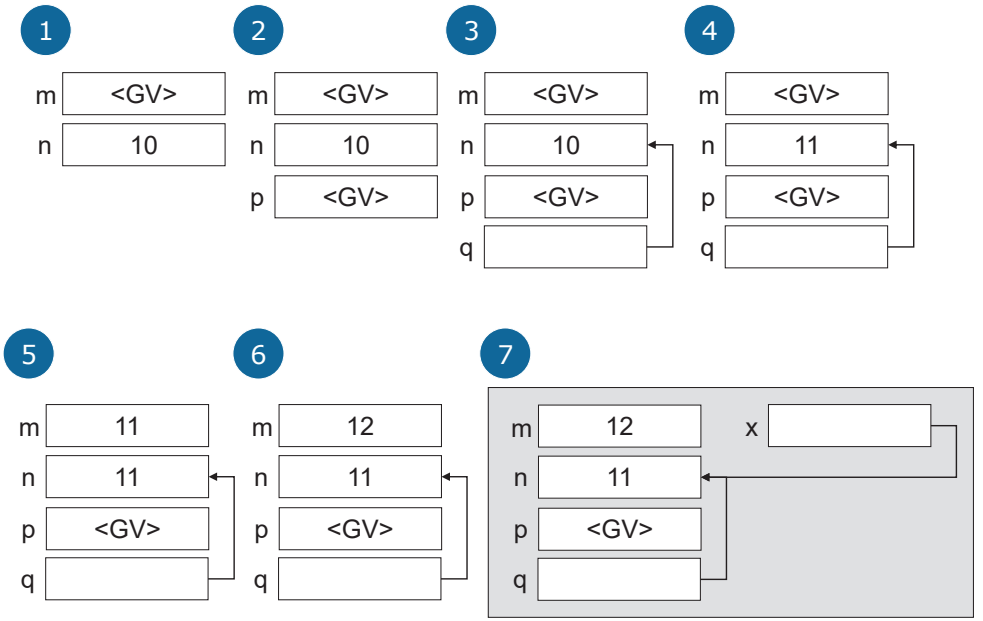

Question 10

- What is the output of the following program?



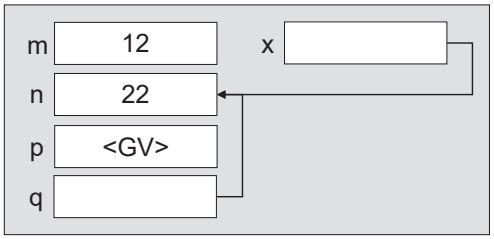
```
#include <iostream>
using namespace std;

int* func(int* x) {
   *x *= 2;
   *x += 1;
   return x;
}

int main() {
   int m, n=10;
   int* p;
   int* q = &n;
   ++(*q);
   m = *q;
   m++;
   p = func(q);
   *p -= 5;
   *q += 5;
   cout << n << " " << m << endl;
   cout << *p << " " << *q << endl;
   return 0;
}
```

Question 10

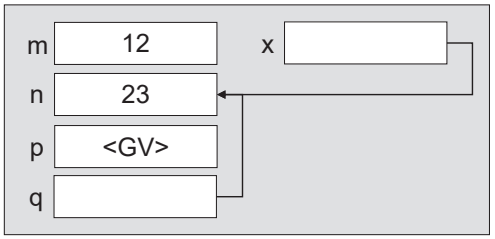


Question 10

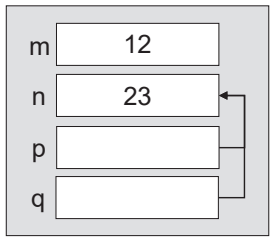
8



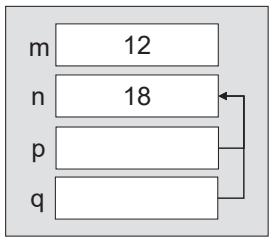
9



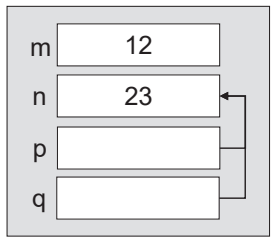
10



11

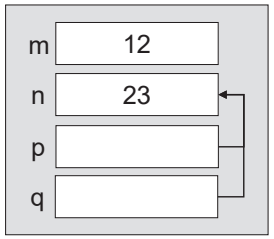


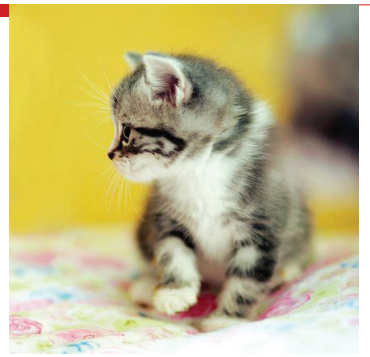
12



Question 10

13





```
cout << n << " " << m << endl;
cout << *p << " " << *q << endl;
```

```
23 12
23 23
```

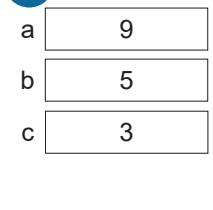

Question 11

- What is the output of the following program?

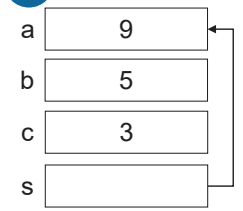
```
#include <iostream>
using namespace std;

int main() {
   int a = 9, b = 5, c = 3;
   int* s = &a;
   int* p = &b;
   int* r = &c;
   (*p)--;
   p = s;
   *p = 2;
   *r = *p + *s + 2;
   s = r;
   r = &b;
   cout << a << b << c << endl;
   cout << *p << *r << *s << endl;
   return 0;
}
```


Question 11

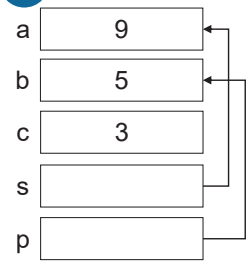
1



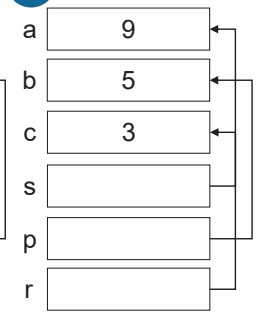
2



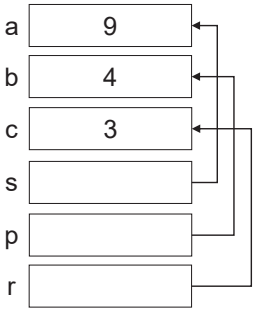
3



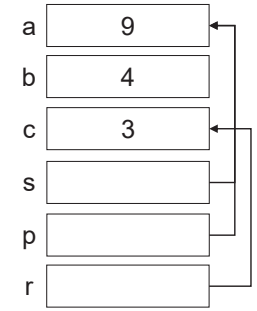
4



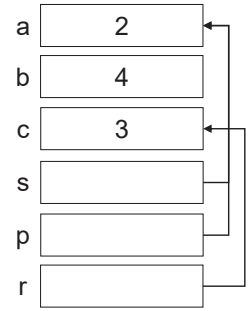
5



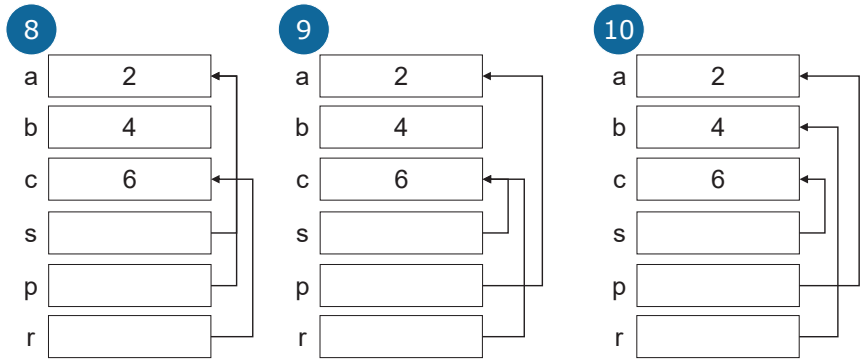
6



7



Question 11




```
cout << a << b << c << endl;
cout << *p << *r << *s << endl;
```

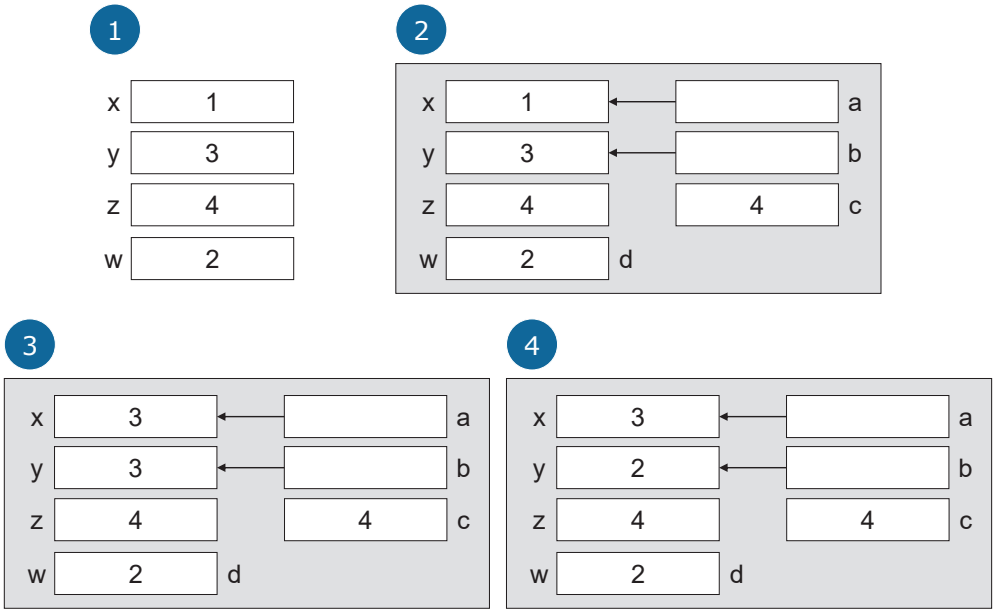
246
246

Question 12

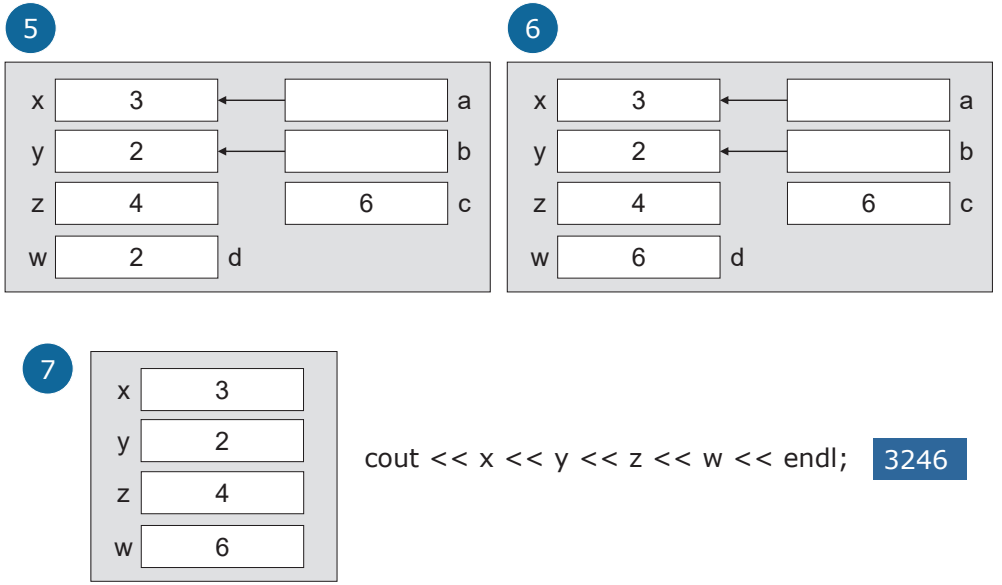
- What is the output of the following program?

```
#include <iostream>
using namespace std;

void fun(int* a, int* b, int c, int& d) {
   *a = *b;
   (*b)--;
   c += 2;
   d *= 3;
}

int main() {
   int x = 1, y = 3, z = 4, w = 2;
   fun(&x, &y, z, w);
   cout << x << y << z << w << endl;
   return 0;
}
```

Question 12



Question 12



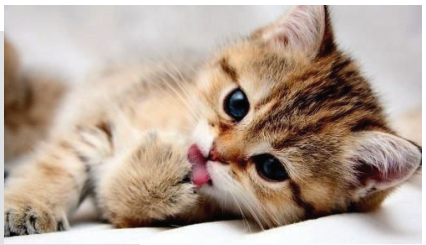

```
cout << x << y << z << w << endl;
```

3246

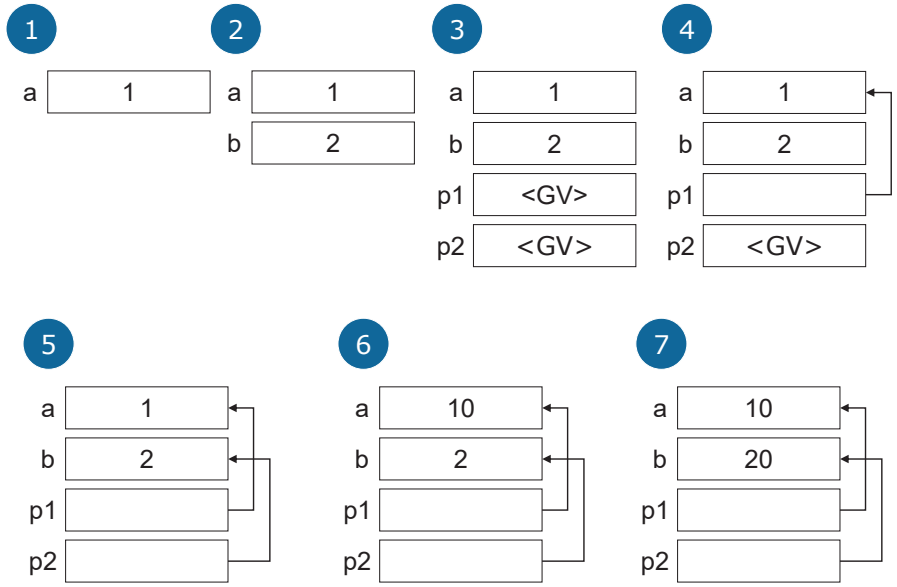
Question 13

a) What is the output produced by the following code?

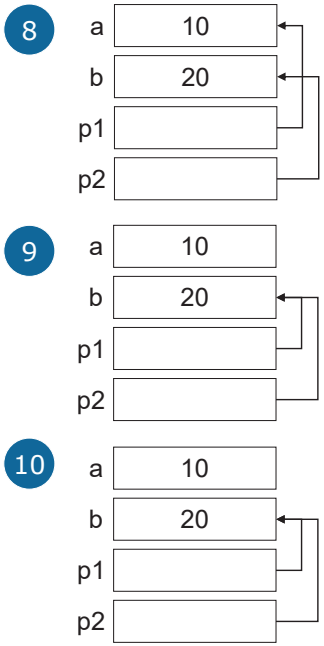
```
int a = 1;
int b = 2;
int* p1, *p2;
p1 = &a;
p2 = &b;
*p1 = 10;
*p2 = 20;
cout << *p1 << " " << *p2 << endl;
p1 = p2;
cout << *p1 << " " << *p2 << endl;
*p1 = 30;
cout << *p1 << " " << *p2 << endl;
```



Question 13



Question 13



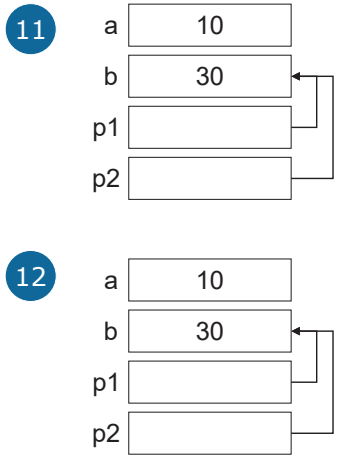
cout << *p1 << " " << *p2 << endl;

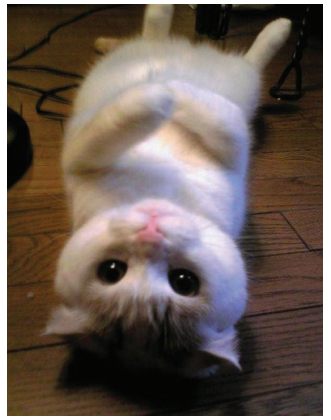
10 20

cout << *p1 << " " << *p2 << endl;

10 20
20 20

Question 13





cout << *p1 << " " << *p2 << endl;

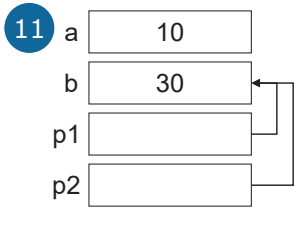
10 20
20 20
30 30

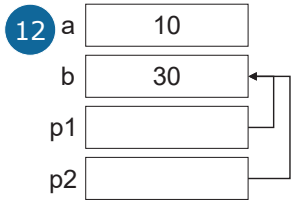
Question 13

b) How would be the output change if you were to replace
*p1 = 30;
with the following
*p2 = 30;

```
int a = 1;
int b = 2;
int* p1, *p2;
p1 = &a;
p2 = &b;
*p1 = 10;
*p2 = 20;
cout << *p1 << " " << *p2 << endl;
p1 = p2;
cout << *p1 << " " << *p2 << endl;
*p1 = 30; *p2 = 30;
cout << *p1 << " " << *p2 << endl;
```

Question 13





cout << *p1 << " " << *p2 << endl;

10 20
20 20
30 30

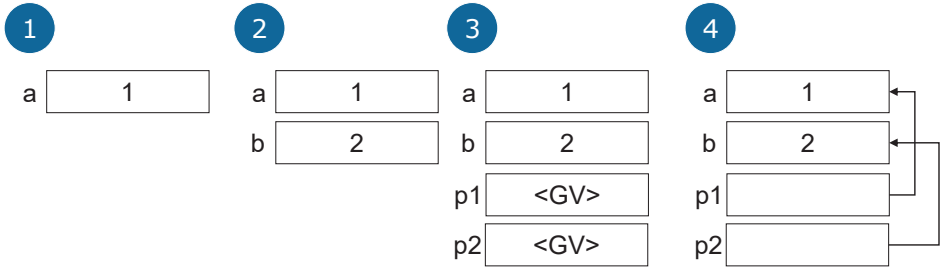
No change! The same as in part (a)

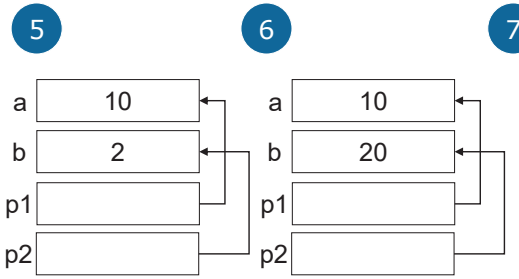
Question 14

- What is the output produced by the following code?

```
int a = 1;
int b = 2;
int* p1, *p2;
p1 = &a;
p2 = &b;
*p1 = 10;
*p2 = 20;
cout << *p1 << " " << *p2 << endl;
*p1 = *p2;
cout << *p1 << " " << *p2 << endl;
*p1 = 30;
cout << *p1 << " " << *p2 << endl;
```

Question 14

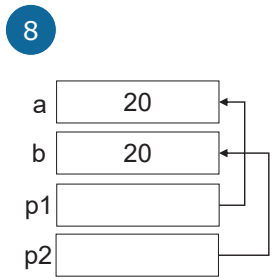




cout << *p1 << " " << *p2 << endl;

10 20

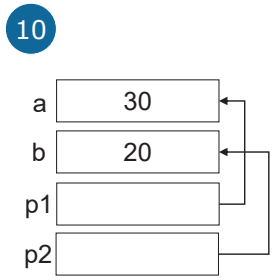
Question 14



9

```
cout << *p1 << " " << *p2 << endl;
```

```
10 20
20 20
```



11

```
cout << *p1 << " " << *p2 << endl;
```

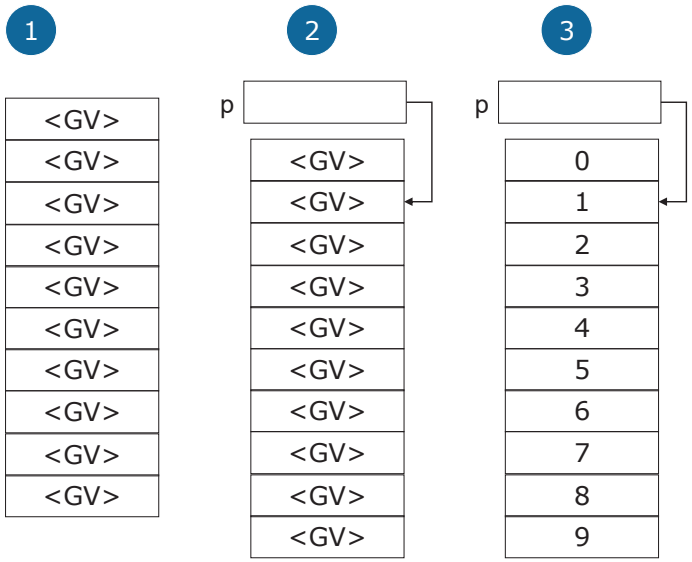
```
10 20
20 20
30 20
```

Question 15

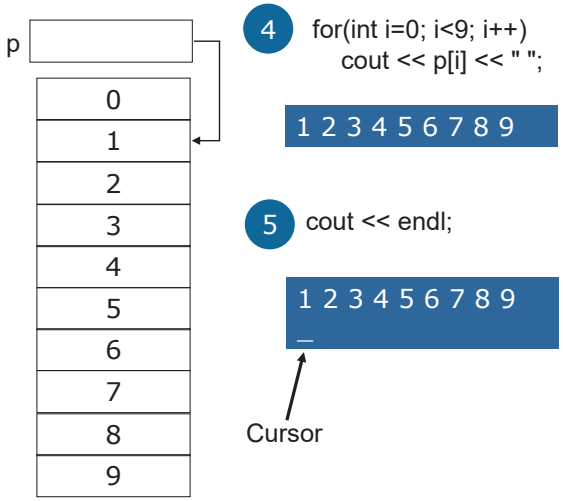
- What is the output of the following code segment? The code is assumed to be embedded in a correct and complete program.

```
int a[10];
int* p = &a[1];
for(int i=0; i<10; i++)
   a[i] = i;
for(int i=0; i<9; i++)
   cout << p[i] << " ";
cout << endl;
```

Question 15



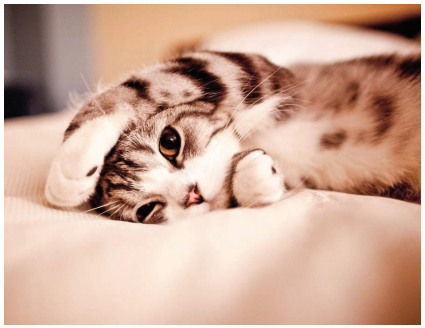
Question 15



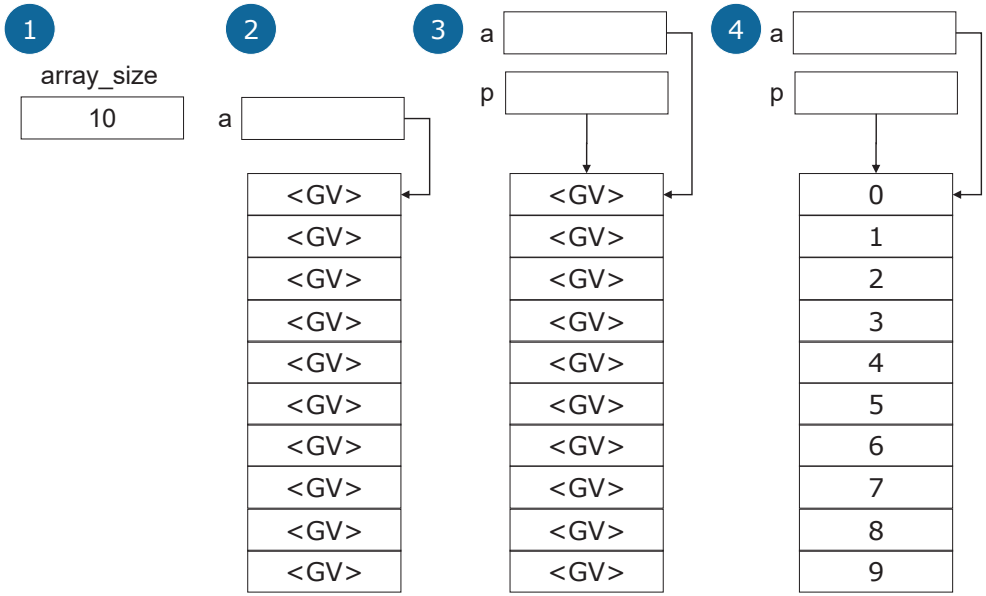
Question 16

- What is the output of the following code segment? The code is assumed to be embedded in a correct and complete program.

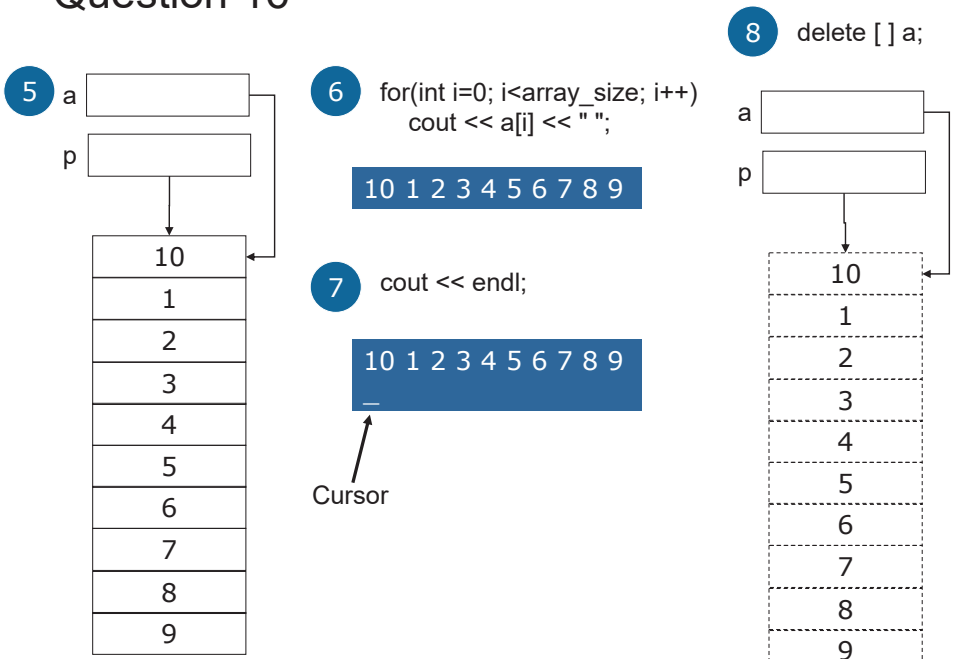
```
int array_size = 10;
int* a = new int[array_size];
int* p = a;
for(int i=0; i<array_size; i++)
   a[i] = i;
p[0] = 10;
for(int i=0; i<array_size; i++)
   cout << a[i] << " ";
cout << endl;
delete [] a;
```



Question 16



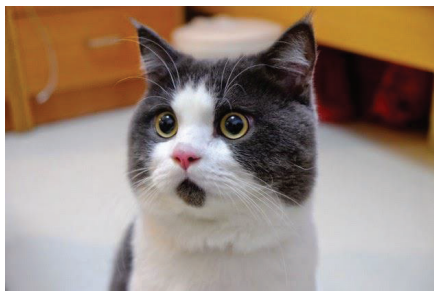
Question 16



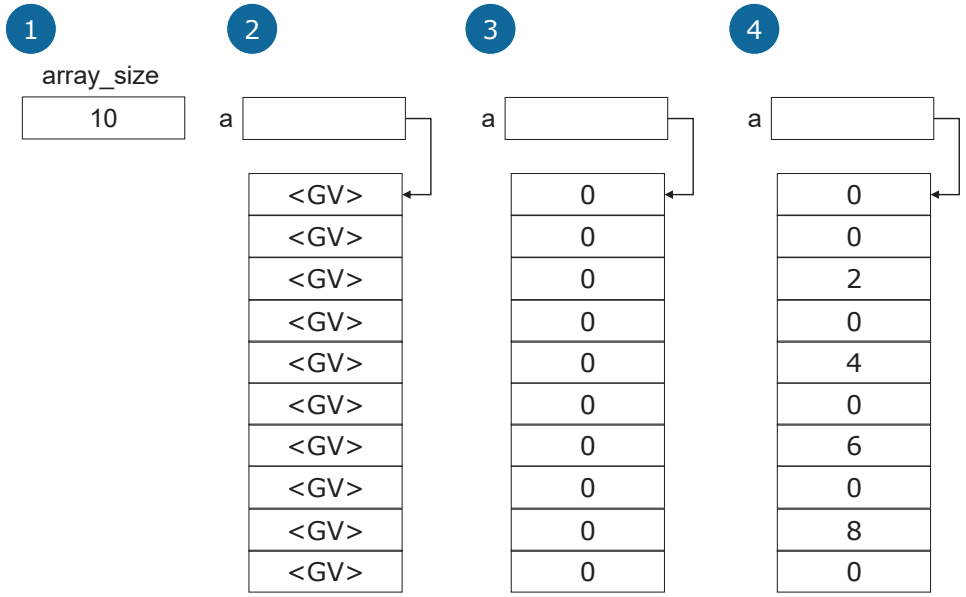
Question 17

- What is the output of the following code segment? The code is assumed to be embedded in a correct and complete program.

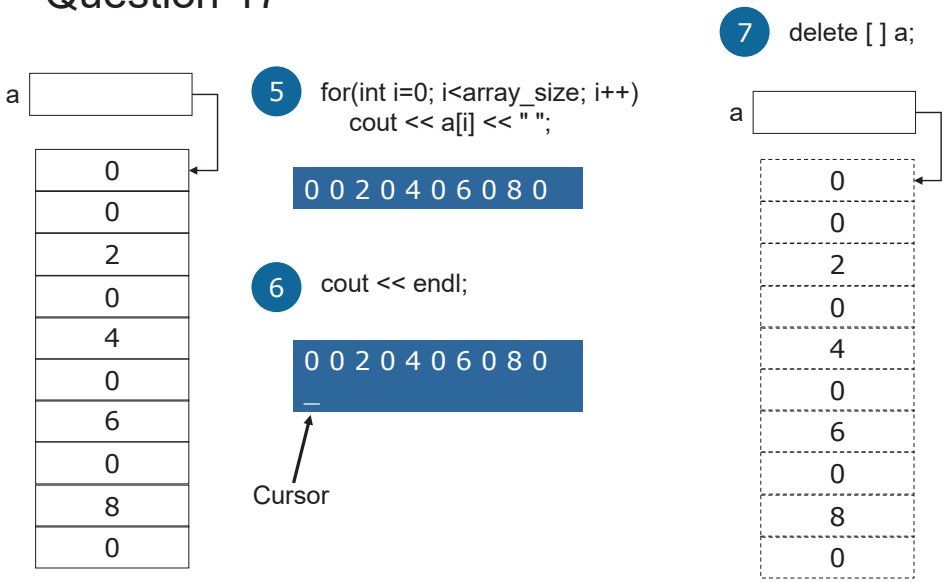
```
int array_size = 10;
int* a = new int[array_size];
for(int i=0; i<array_size; i++)
   a[i] = 0;
for(int i=0; i<array_size; i+=2)
   *(a+i) = i;
for(int i=0; i<array_size; i++)
   cout << a[i] << " ";
cout << endl;
delete [] a;
```



Question 17



Question 17

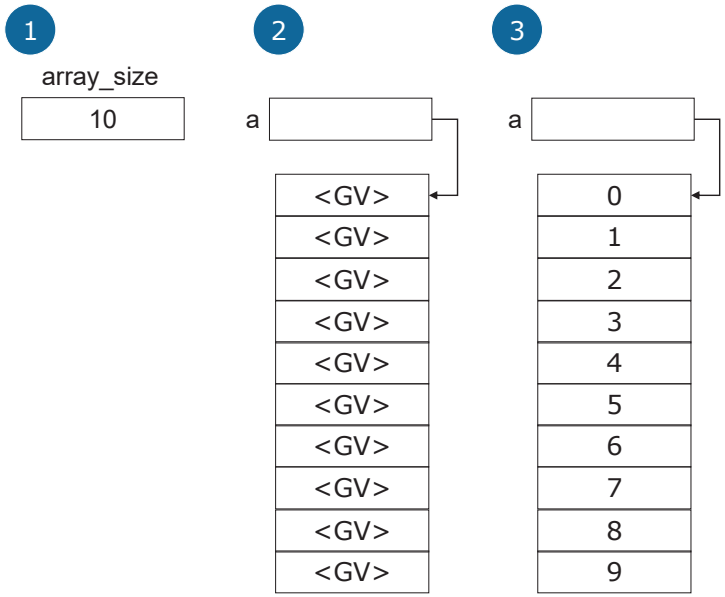


Question 18

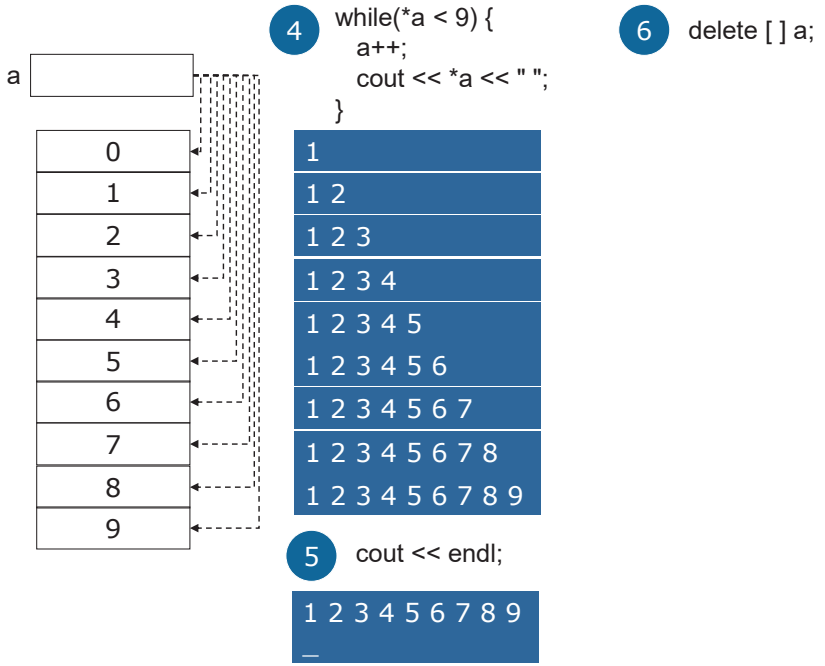
- What is the output of the following code segment? The code is assumed to be embedded in a correct and complete program.

```
int array_size = 10;
int* a = new int[array_size];
for(int i=0; i<array_size; i++)
   a[i] = i;
while(*a < 9) {
   a++;
   cout << *a << " ";
}
cout << endl;
delete [] a;
```

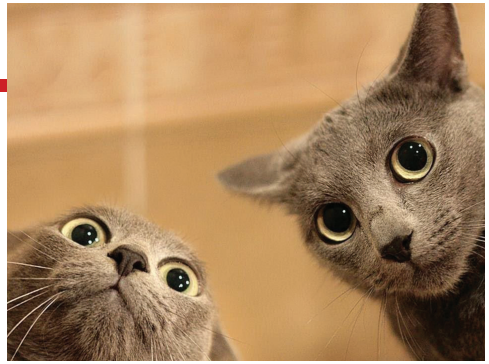
Question 18



Question 18



Question 19



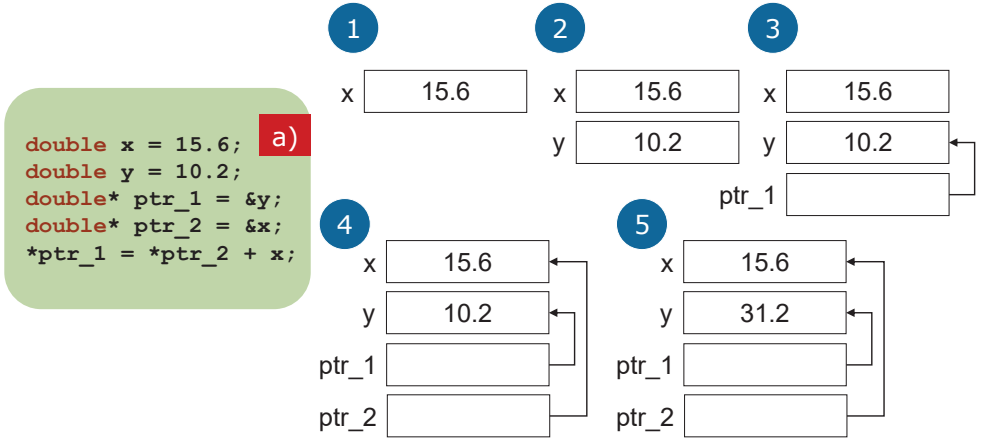
- Suppose a dynamic variable were created as follows:
- char* p;
  p = new char;

Assuming that the value of the pointer variable p has not changed (so it still points to the same dynamic variable), how can you destroy this new dynamic variable and return the memory it uses to the operating system so that the memory can be reused.

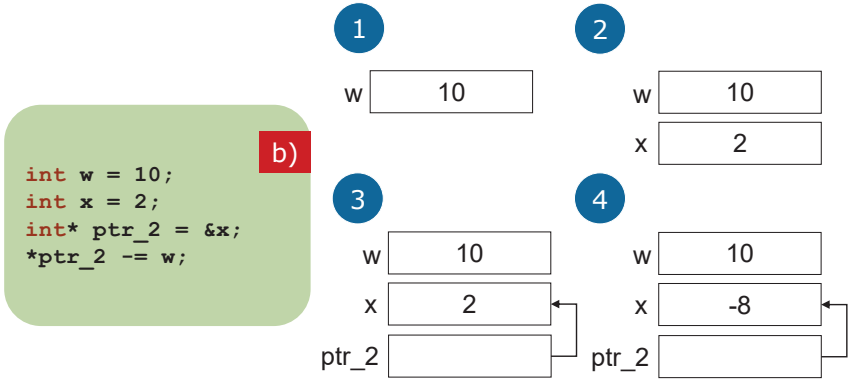
Answer: delete p;

Question 20

- For each of the problems that follow, give a memory snapshot that includes all objects after the problem statements are executed. Include as much information as possible. Use question marks to indicate memory locations that have not been initialized.



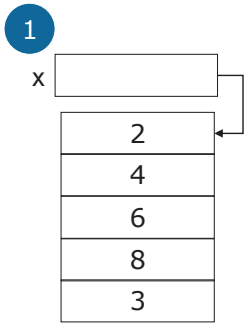
Question 20

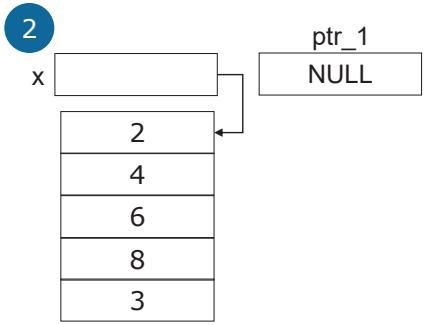


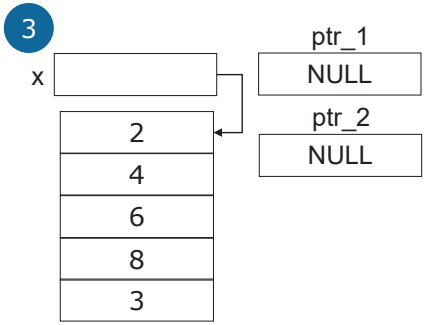
Question 20

```
int x[5] = { 2, 4, 6, 8, 3 };
int* ptr_1 = NULL;
int* ptr_2 = NULL;
int* ptr_3 = NULL;
ptr_3 = &x[0];
ptr_1 = ptr_3 + 2;
ptr_2 = ptr_3 + 2;
```

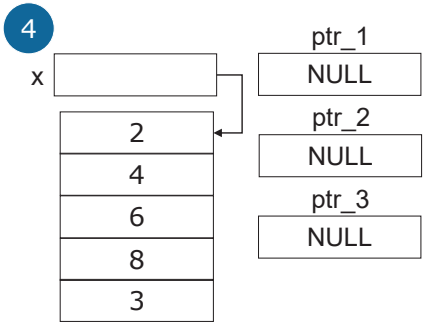
c)

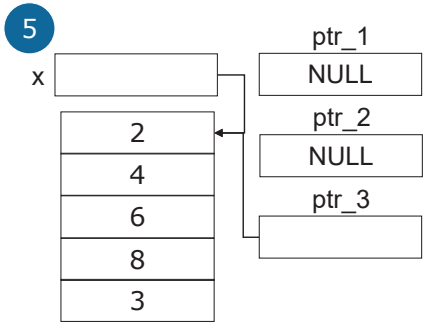


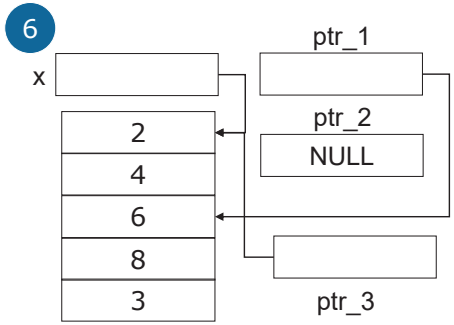


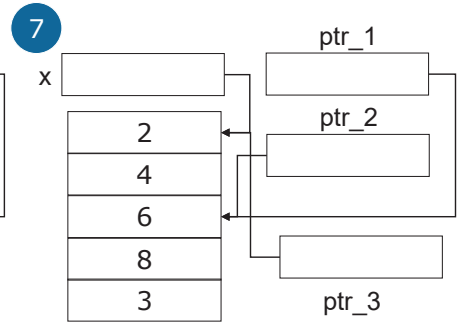


Question 20





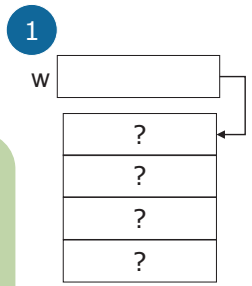


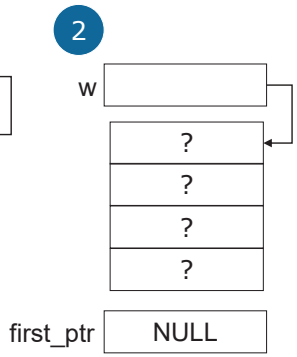


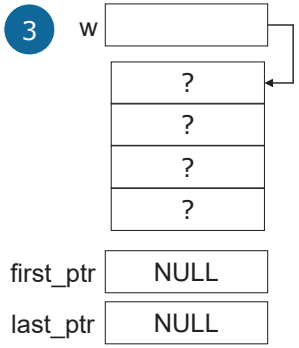
Question 20

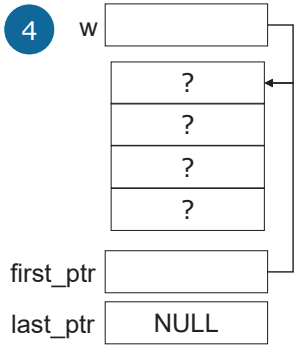
```
int w[4];
int* first_ptr = NULL;
int* last_ptr = NULL;
first_ptr = w;
last_ptr = first_ptr + 3;
```

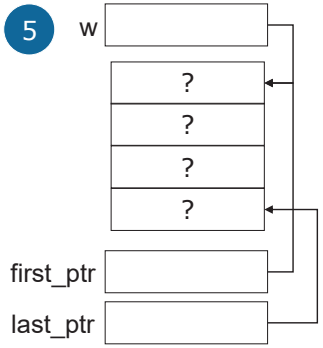
d)











Question 21

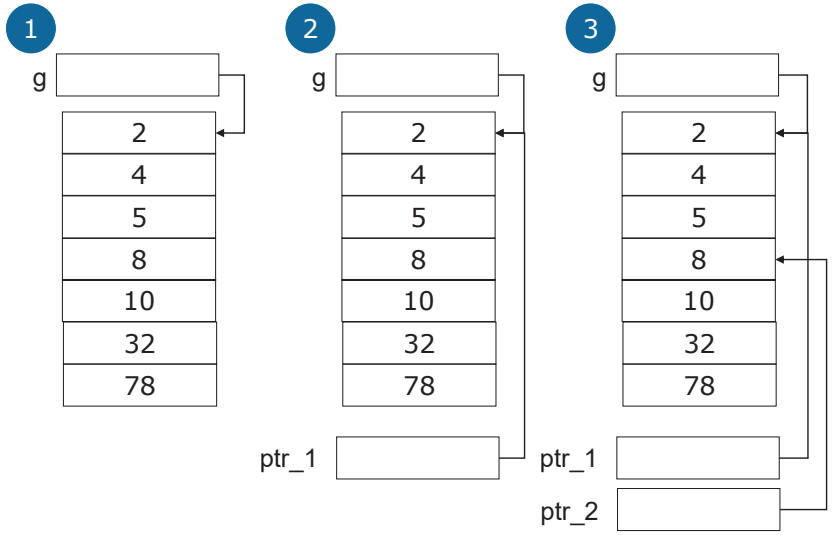
- Assume that an array g is defined with the following statement:

```
int g[] = { 2, 4, 5, 8, 10, 32, 78 };
int* ptr1 = g;
int* ptr2 = &g[3];
```

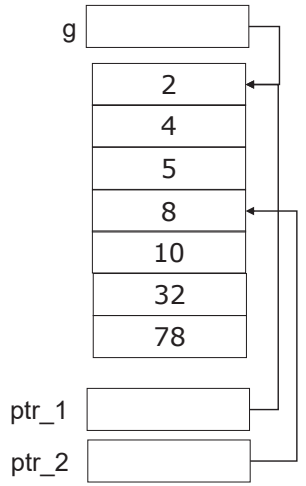
- Give a memory snapshot, including the array values. Using this information, give the value of the following statements.

a) *g
b) *(g+1)
c) *g + 1
d) *(g + 5)
e) *ptr1
f) *ptr2
g) *(ptr1 + 1)
h) *(ptr2 + 2)

Question 21



Question 21



a) *g 2

b) *(g+1) 4

c) *g + 1 3

d) *(g + 5) 32

e) *ptr1 2

f) *ptr2 8

g) *(ptr1 + 1) 4

h) *(ptr2 + 2) 32

Question 22

- Assume that following variables have been defined with the following statements:

```
int i = 5;
int j = 10;
int* iPtr = &i;
const int* cPtr = &j;
int* const jPtr = &j;
```

Determine whether the following statements are valid or invalid.

a) cPtr = jPtr; Valid

b) jPtr = cPtr; Invalid

c) *cPtr = *jPtr; Invalid

d) *cPtr = *iPtr; Invalid

e) *jPtr = *cPtr; Valid

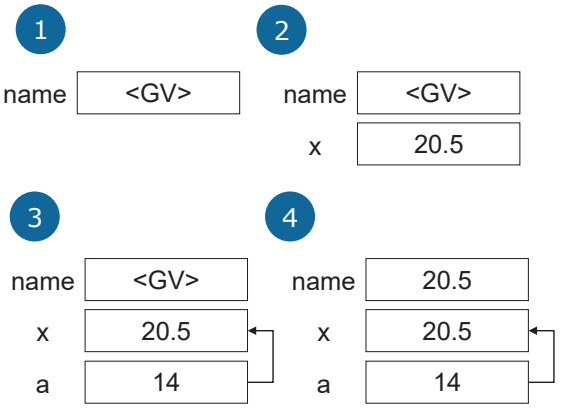
f) iPtr = cPtr; Invalid

Question 23

- Give the corresponding snapshot of memory after the following set of statements is executed:

```
double name;
double x = 20.5;
double* a = &x;
name = *a;
```

- Assume the address of name is 10 and the address of x is 14



Question 24

- Give the following statements:

```
int i1, i2;
int* p1, *p2;
i1 = 5;
p1 = &i1;
i2 = *p1/2 + 10;
p2 = p1;
```

1

i1	<GV>
i2	<GV>

2

i1	<GV>
i2	<GV>
p1	<GV>
p2	<GV>

3

i1	5
i2	<GV>
p1	<GV>
p2	<GV>

4

i1	5
i2	<GV>
p1	
p2	<GV>

5

i1	5
i2	12
p1	
p2	<GV>

6

i1	5
i2	12
p1	
p2	

Question 24

a) What is the value of i1? 5
b) What is the value of i2? 12
c) What is the value of *p1? 5
d) What is the value of *p2? 5

i1	5
i2	12
p1	
p2	

Question 25

- What unfortunate misinterpretation can occur with the following declaration?

int* int_ptr1, int_ptr2;

An unfortunate misinterpretation of the above declaration would be treating int_ptr2 as integer pointer variable, while it is not. It is actually an integer variable

Question 26

- Give three uses of the * operator. State what the * is doing, and name the use of the * that you present.

1. Multiplication, * in this example is an multiplication operator
   int a = 10;
   int b = 20;
   int c = a * b;
2. Pointer variable declaration, * sticks next to a data type for declaration of variable
   int* a;
3. De-referencing, * sticks to a pointer variable refers to the variable that the pointer points to
   int a = 10;
   int* b = &a;
   *b = 20; // Equivalent to assigning 20 to variable a

Question 27

- What are the two common pointer errors made by many C++ programmers? Present an example for each of these common errors.
  1. Memory leakage
     int* a = new int;
     a = new int;
     // variable a now points to the newly created int, while the
     // address of the old one is lost and not yet returned to OS
  2. Dangling pointer
     int* a = new int;
     int* b = a;
     delete a;
     a = NULL;
     // variable b still stores the address of the dynamic variable, which has been returned back to OS using statement: delete a

Question 28

What is the output of the following program?

```
#include <iostream>
using namespace std;

void figure_me_out(int& x, int y, int& z);

int main() {
     int a, b, c;
     a = 10;
     b = 20;
     c = 30;
     figure_me_out(a, b, c);
     cout << a << " " << b << " " << c;
     return 0;
}

void figure_me_out(int& x, int y, int& z) {
     cout << x << " " << y << " " << z << endl;
     x = 1;
     y = 2;
     z = 3;
     cout << x << " " << y << " " << z << endl;
}
```

Question 28

1

a	<GV>
b	<GV>
c	<GV>

2

a	10
b	20
c	30

3

x a	10	y	20
b	20		
z c	30		

4 cout << x << " " << y << " " << z << endl;

```
10 20 30
_
```

5

x a	1	y	2
b	20		
z c	3		

6 cout << x << " " << y << " " << z << endl;

```
10 20 30
1 2 3
_
```

a	1
b	20
c	3

7 cout << a << " " << b << " " << c << endl;

```
10 20 30
1 2 3
1 20 3
```