Assignment 1

What is printed out? Are there any problems (errors)?

int a = 3;
int *b = &a;

cout << b << endl; // output the address of a.
cout << *b << endl; // output the value at the address that b is pointing to (value of a : 3).
cout << &b << endl; // output the address of pointer b.

cout << a << endl; // output the value of a is 3.
cout << &a << endl; // output the address of a.
=> no error.

Assignment 2

```
int x, z;
float y;
char ch, * chp;
int* ip1, * ip2;
float* fp;

x = 100;
y = 20.0;
z = 50;
ch = 'Z';

ip1 = &x;  // the ip1 pointer points to x.
ip2 = &z; // the ip2 pointer points to z.
fp = &y; // the fp pointer points to y.
chp = &ch; // the chp pointer points to ch.

ip2 = ip1; // specify ip2 to point to where ip1 is pointing, now both ip2 and ip1 pointers hold the address of x.
ip1 = &z; // the ip1 pointer points to z.
*ip1 = *ip2; // assign the value pointed by ip2 to the value pointed by ip1 (z = 100 )

*ip1 = 200; // as z = 200.
*ip1 = *ip2 + 300; // as z = 100 + 300 = 400;
*fp = 1.2; // as y =1.2

cout << x << endl;  // print x = 100;
cout << y << endl;  // print y = 1.2;
cout << z << endl;  // print z = 400;

cout << ip1 << endl; // output the address that the pointer ip1 is pointing to.
cout << *ip1 << endl; // output the value of the address the pointer ip1 is pointing to, 400;
cout << &ip1 << endl; // output the address of the ip1 pointer.

cout << ip2 << endl; // output the address the ip2 pointer is pointing to.
cout << *ip2 << endl; // output the value of the address the ip2 pointer is pointing to, output 100;
cout << &ip2 << endl; // output the address of the ip2 pointer.

cout << fp << endl; // Outputs the address the fp pointer is pointing to.
cout << *fp << endl; // output the value at the address that the fp pointer is pointing to, output 1.2;
cout << &fp << endl; // output the address of the pointer fp.

cout << chp << endl; // Outputs the address that the chp pointer is pointing to.
cout << *chp << endl; // output the value at the address that the pointer chp is pointing to, output Z;
cout << &chp << endl; // output the address of the chp pointer.
```

=> no error

Assignment 3

What is printed out? Are there any problems (errors)?

```
int *a = new int; // create an “unnamed” variable and assign a points to it
int *b = new int; // create an “unnamed” variable and assign b points to it
*a = 2; // assign the value 2 to the allocated variable.
b = a; // b to points to the variable that a is pointing to.
cout << *a << endl; // output 2
cout << *b << endl; // output 2.
delete a;
delete b;
```

=> no error.

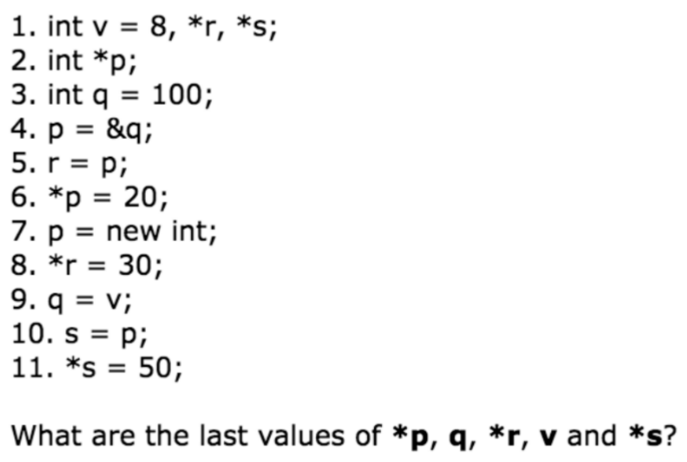
Assignment 4

What is printed out? Are there any problems (errors)?

int a = 3;
int *p = &a; // p points to a.
cout << *p << endl; // output the value at the address that pointer p is pointing to, 3
p = new int(5); // creates a dynamic allocated variable, points the pointer p to that variable, and initializes the value at that variable to 5.
cout << *p << endl; // output the value at that variable is 5.

=> no error

Assignment 5



```
1. int v = 8, *r, *s;
2. int *p;
3. int q = 100;
4. p = &q;
5. r = p;
6. *p = 20;
7. p = new int;
8. *r = 30;
9. q = v;
10. s = p;
11. *s = 50;
```

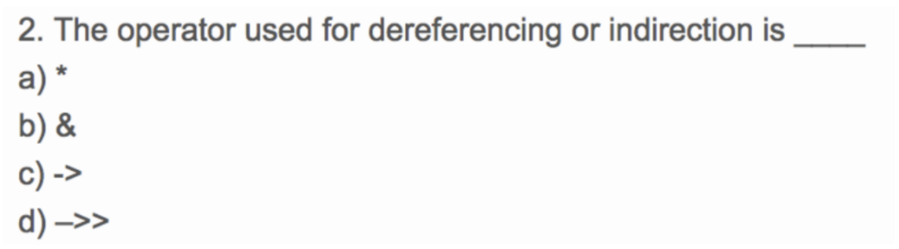
What are the last values of ***p**, **q**, ***r**, **v** and ***s**?

Dòng 1, 2, 3: initialize variables, pointers.
Dòng 4: p points to q.
Dòng 5: pointer r to points to where the pointer p is pointing to.
Dòng 6: assign the value at the address that pointer p is pointing to = 20; means q = 20
Dòng 7: creates a new dynamic allocated variable and assigns p points to it.
Dòng 8: assign the value at the address that pointer r is pointing to = 30; means q = 30;
Dòng 9: q = 8, now r points to q so the value of *r = 8.
Dòng 10: pointer r points to where the pointer p is pointing to.
Dòng 11: assigns the value at the address that pointer s is pointing to is 50.

=> **<u>result</u>**: *p == 50, q= = 8, *r == 8, v == 8, *s == 50

=> no error.

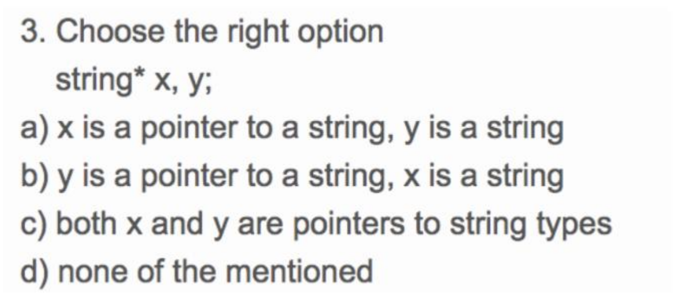
Assignment 13



2. The operator used for dereferencing or indirection is _____
a) *
b) &
c) ->
d) –>>

=> a.
Explanation: the * operator is used to read the value of the address that the pointer points to.

Assignment 14



3. Choose the right option

string* x, y;

a) x is a pointer to a string, y is a string

b) y is a pointer to a string, x is a string

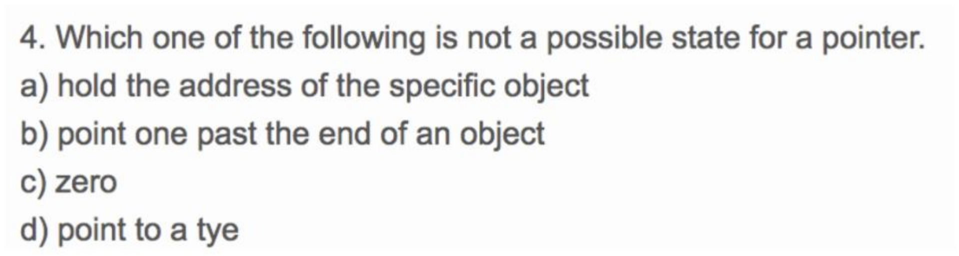
c) both x and y are pointers to string types

d) none of the mentioned

=> a

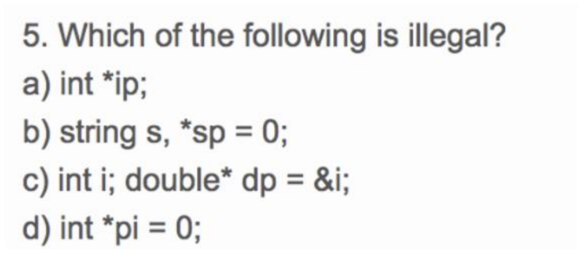
Explanation: if initializing like this, only variable x is a pointer and y has type string => to make both x, y to be pointers => string *x, *y;

Assignment 15


4. Which one of the following is not a possible state for a pointer.
a) hold the address of the specific object
b) point one past the end of an object
c) zero
d) point to a tye

=> d
Explanation: pointer stores the address of a particular variable.

Assignment 16


5. Which of the following is illegal?

a) int *ip;

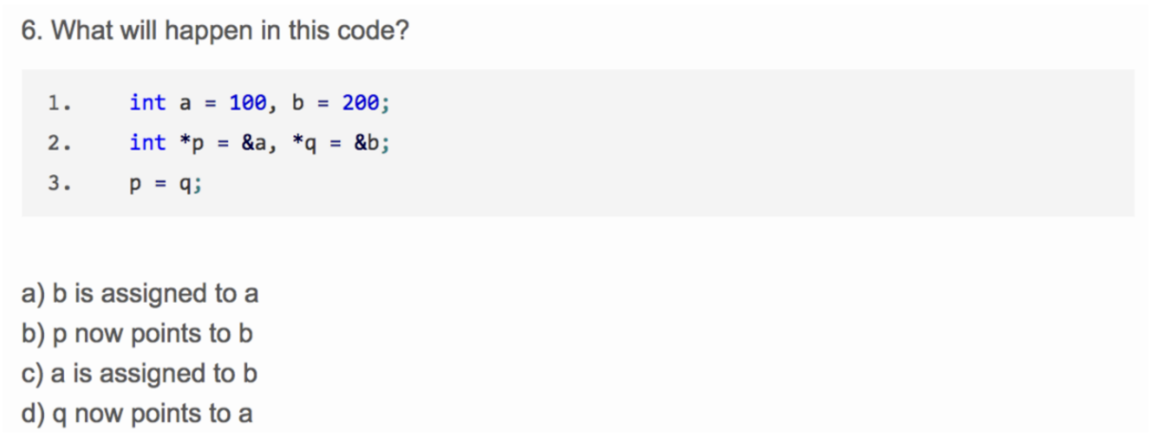
b) string s, *sp = 0;

c) int i; double* dp = &i;

d) int *pi = 0;

=> c
<u>**Explanation**</u>: because i is an integer type, and the pointer dp is a double type.
=> int i; int* dp = &i; || int i; double* dp = (double*)&i;

Assignment 17



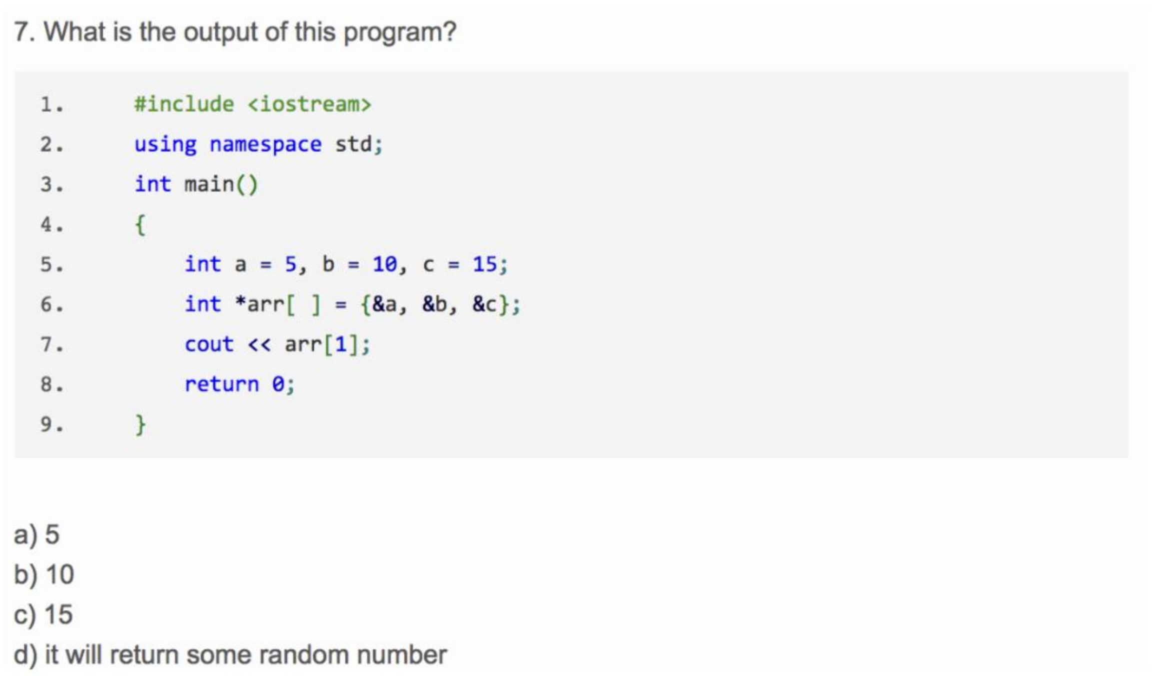
6. What will happen in this code?

```
int a = 100, b = 200;
int *p = &a, *q = &b;
p = q;
```

a) b is assigned to a
b) p now points to b
c) a is assigned to b
d) q now points to a

=> b
<u>**Explanation**</u>: the command p =q is assigning that p points to where q is pointing to, q is pointing to b, so p points to b.

Assignment 18



7. What is the output of this program?

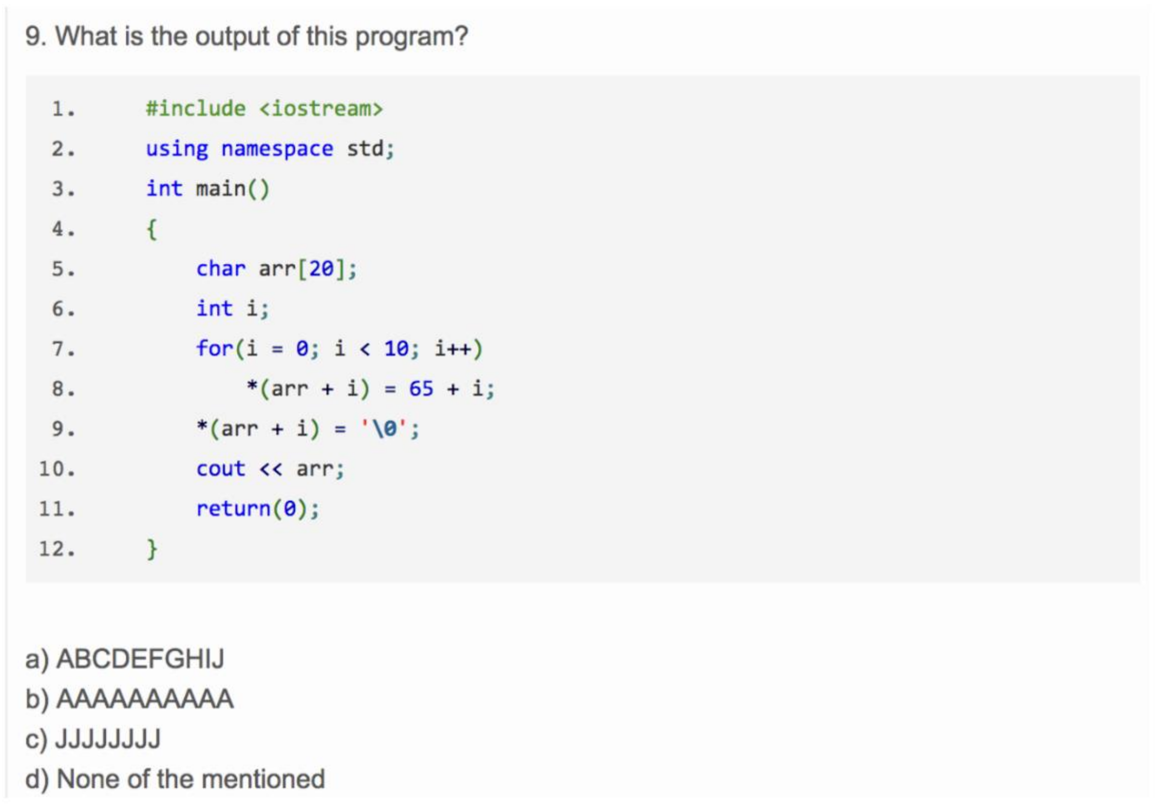
```
#include <iostream>
using namespace std;
int main()
{
    int a = 5, b = 10, c = 15;
    int *arr[ ] = {&a, &b, &c};
    cout << arr[1];
    return 0;
}
```

a) 5
b) 10
c) 15
d) it will return some random number

=> d

Explanation: the first line in main function initializes 3 variables a,b,c with values of 5,10,15 respectively. The second line initializes an array of pointers and passes to it the elements that are the addresses of a, b, c. In the command cout << arr[1] is to print the value of the second element of the array which is the address of b, at each time the program runs, the address of the variables will change.

Assignment 19



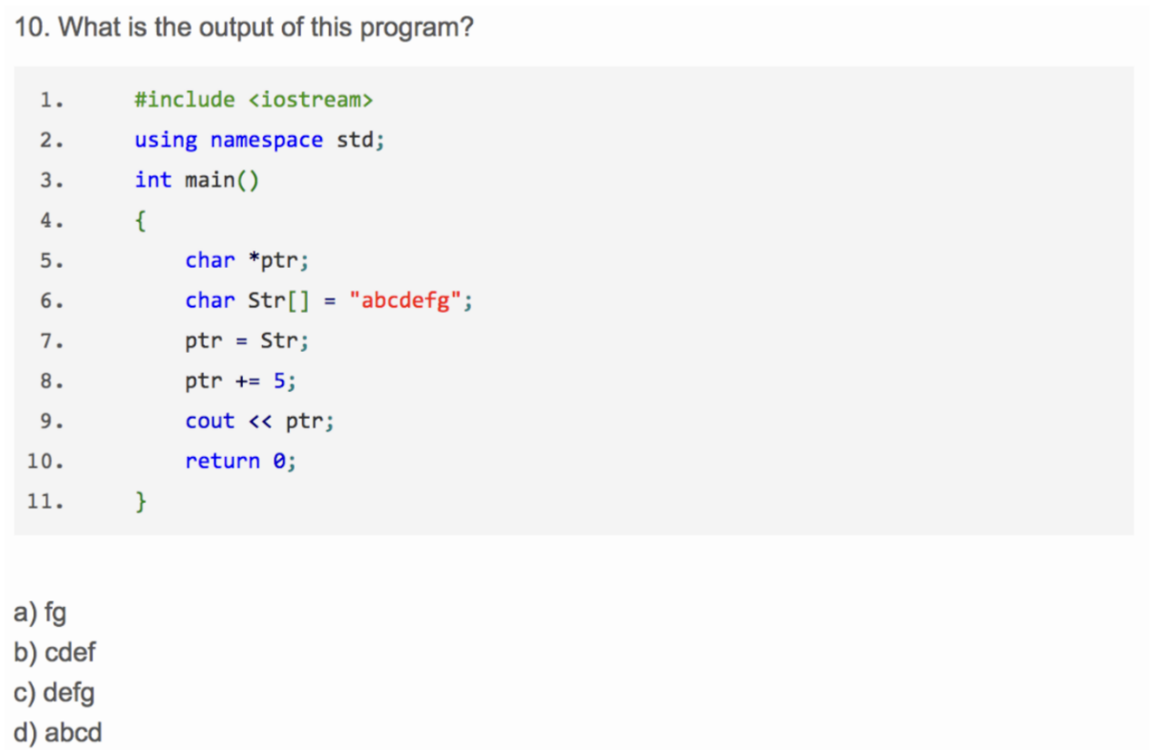
9. What is the output of this program?

```
 1.     #include <iostream>
 2.     using namespace std;
 3.     int main()
 4.     {
 5.         char arr[20];
 6.         int i;
 7.         for(i = 0; i < 10; i++)
 8.             *(arr + i) = 65 + i;
 9.         *(arr + i) = '\0';
10.         cout << arr;
11.         return(0);
12.     }
```

a) ABCDEFGHIJ
b) AAAAAAAAAA
c) JJJJJJJ
d) None of the mentioned

=> a
Explanation: the 8th line runs the element of the array starting from 65 + i (0<= i < 10) so according to the ASCII code, increasing i will make characters from A -> J.

Assignment 20

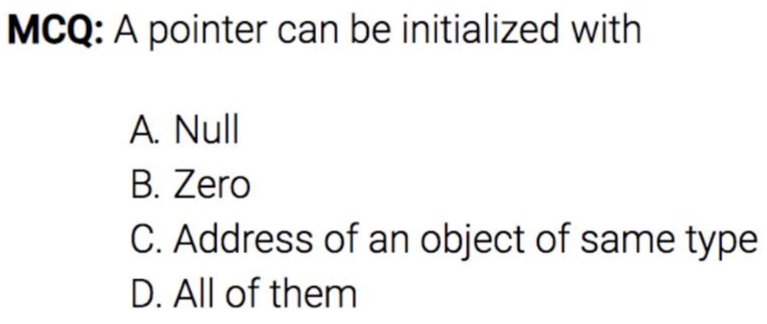

10. What is the output of this program?

```
1.      #include <iostream>
2.      using namespace std;
3.      int main()
4.      {
5.          char *ptr;
6.          char Str[] = "abcdefg";
7.          ptr = Str;
8.          ptr += 5;
9.          cout << ptr;
10.         return 0;
11.     }
```

a) fg
b) cdef
c) defg
d) abcd

=> a
<u>Explanation</u>: pointer ptr is pointing to the first address of the array str[0] string, line 8th pointer ptr+ 5, which means pointer ptr points to str[5] of the string array.

Assignment 21

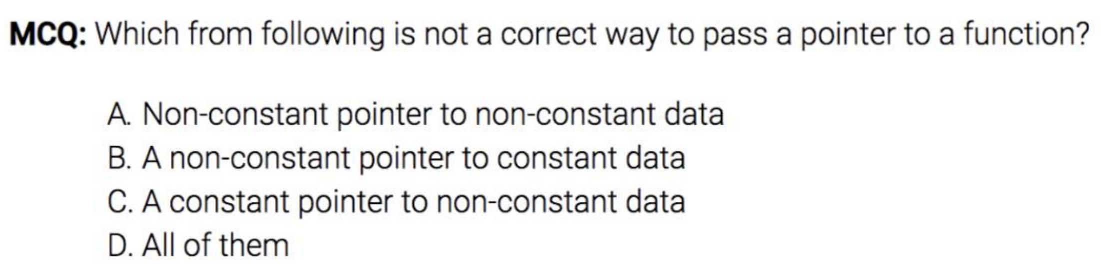


MCQ: A pointer can be initialized with

A. Null
B. Zero
C. Address of an object of same type
D. All of them

=> D
Explanation: pointer can be initialized as a null pointer (pointer isn’t point to any address) also equivalent to assigning initial value to pointer to 0, the address of variable and the pointer must have same type.

Assignment 22

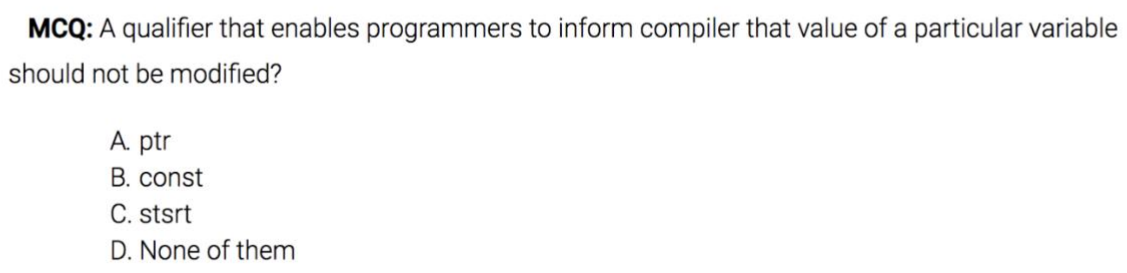


MCQ: Which from following is not a correct way to pass a pointer to a function?

A. Non-constant pointer to non-constant data
B. A non-constant pointer to constant data
C. A constant pointer to non-constant data
D. All of them

=> D
Explanation: the pointer passed to the function has any data type that matches the data type of the passed address.

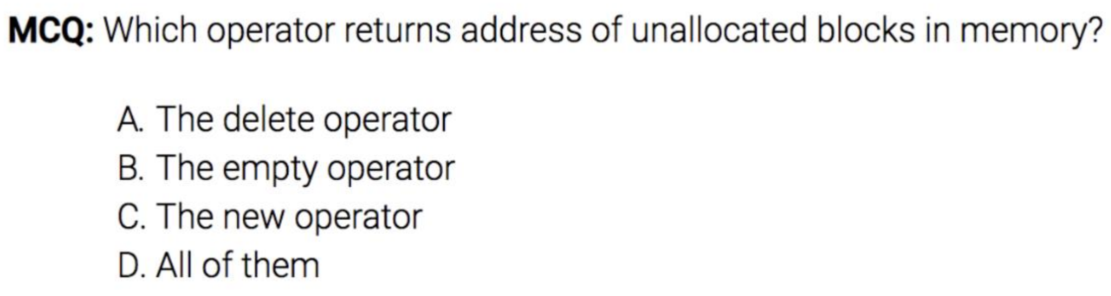
Assignment 23


MCQ: A qualifier that enables programmers to inform compiler that value of a particular variable should not be modified?

A. ptr
B. const
C. stsrt
D. None of them

=> b
Explanation: when using the const keyword before a variable, the value of the variable will not change during program execution.

Assignment 24



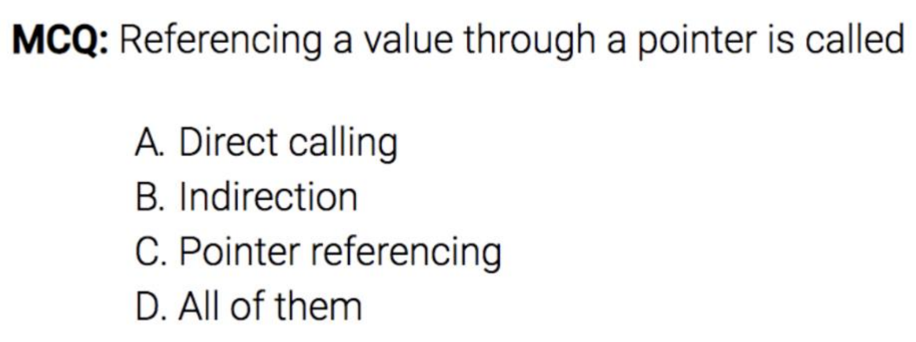
MCQ: Which operator returns address of unallocated blocks in memory?

A. The delete operator
B. The empty operator
C. The new operator
D. All of them

=> c

Explanation: the new operator is used to request memory allocation on the heap whose size may not be fixed.

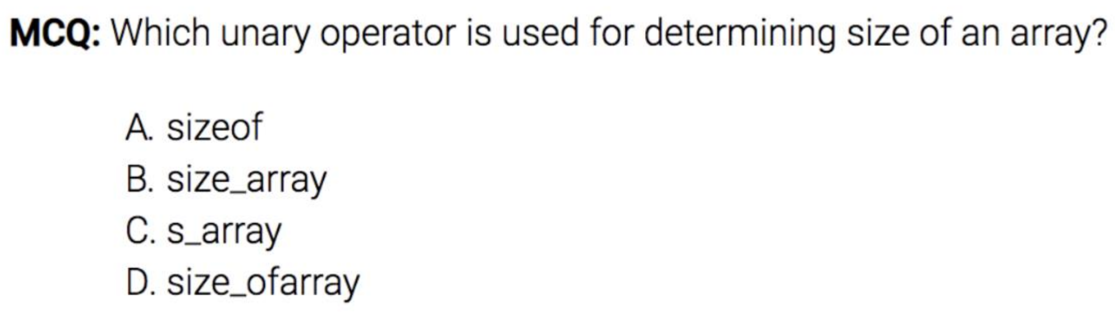
Assignment 25


MCQ: Referencing a value through a pointer is called

A. Direct calling
B. Indirection
C. Pointer referencing
D. All of them

=> b
Explanation: indirection operator (* operator) is used to get the value of an address through a pointer.

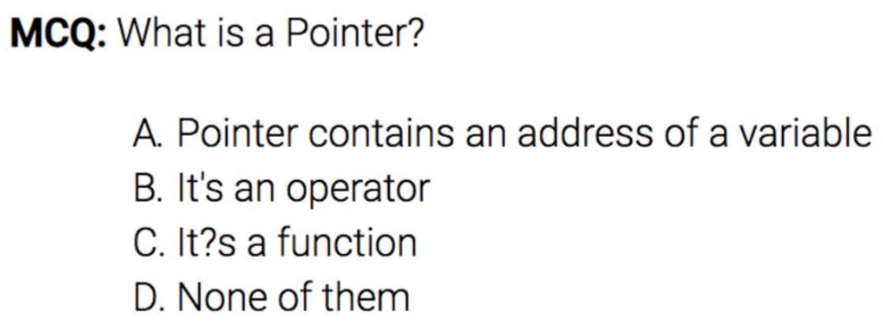
Assignment 26


MCQ: Which unary operator is used for determining size of an array?

A. sizeof
B. size_array
C. s_array
D. size_ofarray

=> a
Explanation: the sizeof operator takes a parameter of any data and returns the size of that data type.

Assignment 27

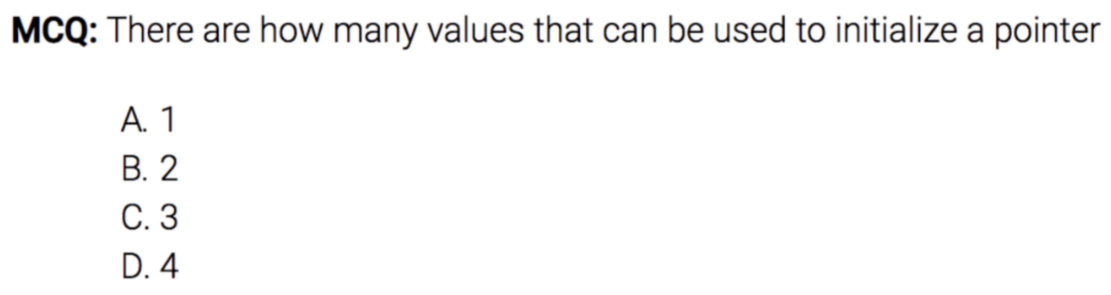


MCQ: What is a Pointer?

A. Pointer contains an address of a variable
B. It's an operator
C. It?s a function
D. None of them

=> a.
Explanation: pointer is a variable whose value is the address of another variable, or it can be said that pointer contains the address of another variable.

Assignment 28

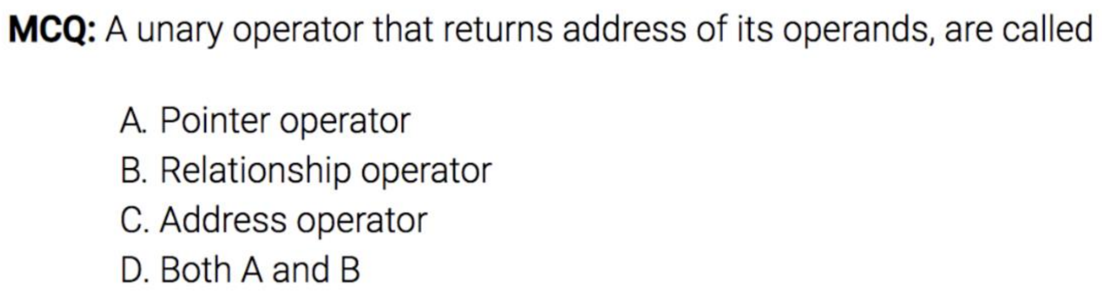


MCQ: There are how many values that can be used to initialize a pointer

A. 1
B. 2
C. 3
D. 4

=> c
Explanation: there are 3 assignable values when initializing a pointer: the address of a variable, null, and 0.

Assignment 29

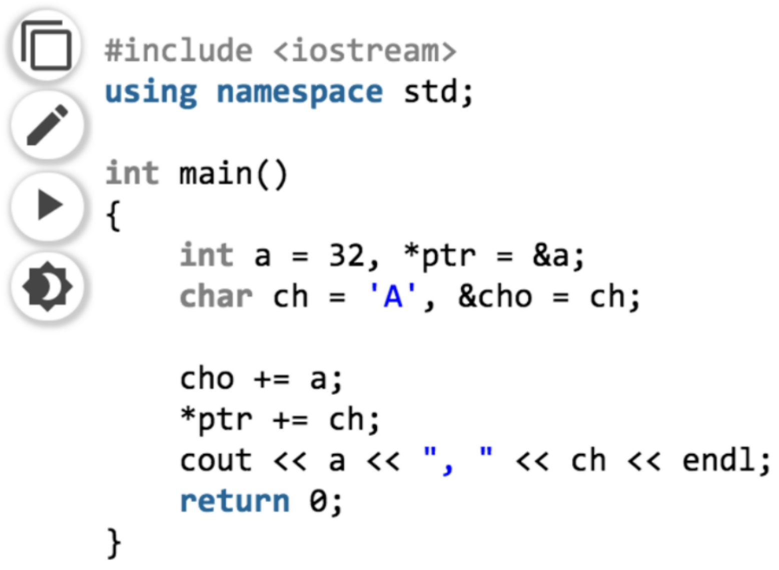

MCQ: A unary operator that returns address of its operands, are called

A. Pointer operator
B. Relationship operator
C. Address operator
D. Both A and B

=> c
Explanation: the address operator (operator "&") used when placing before a variable will get the address of that variable.

Assignment 30

1. What will be the output of the following program?



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

int main()
{
    int a = 32, *ptr = &a;
    char ch = 'A', &cho = ch;

    cho += a;
    *ptr += ch;
    cout << a << ", " << ch << endl;
    return 0;
}
```

Options:

a. 32, A

b. 32, a

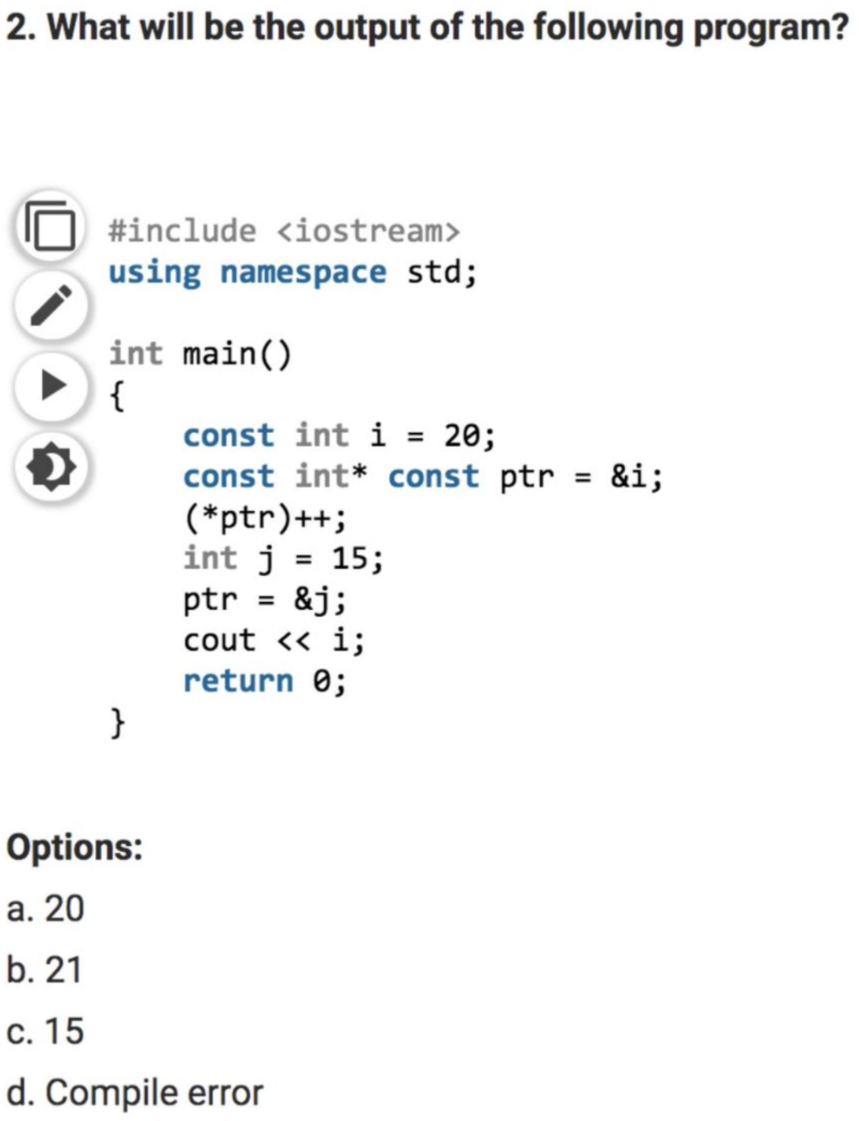
c. 129, a

d. 129, A

=> c

Explanation: the pointer ptr stores the address of the variable a whose value is 32, cho is the reference variable that refers to the variable ch. The command “cho +=a” means increasing the value of ch to 32 according to the ASCII, making the value of ch equal to a. “*ptr +=ch” means taking 32 plus ch gives the value a = 129 according to the ASCII => so the output is 129, a.

Assignment 31



2. What will be the output of the following program?

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

int main()
{
    const int i = 20;
    const int* const ptr = &i;
    (*ptr)++;
    int j = 15;
    ptr = &j;
    cout << i;
    return 0;
}
```

Options:

a. 20

b. 21

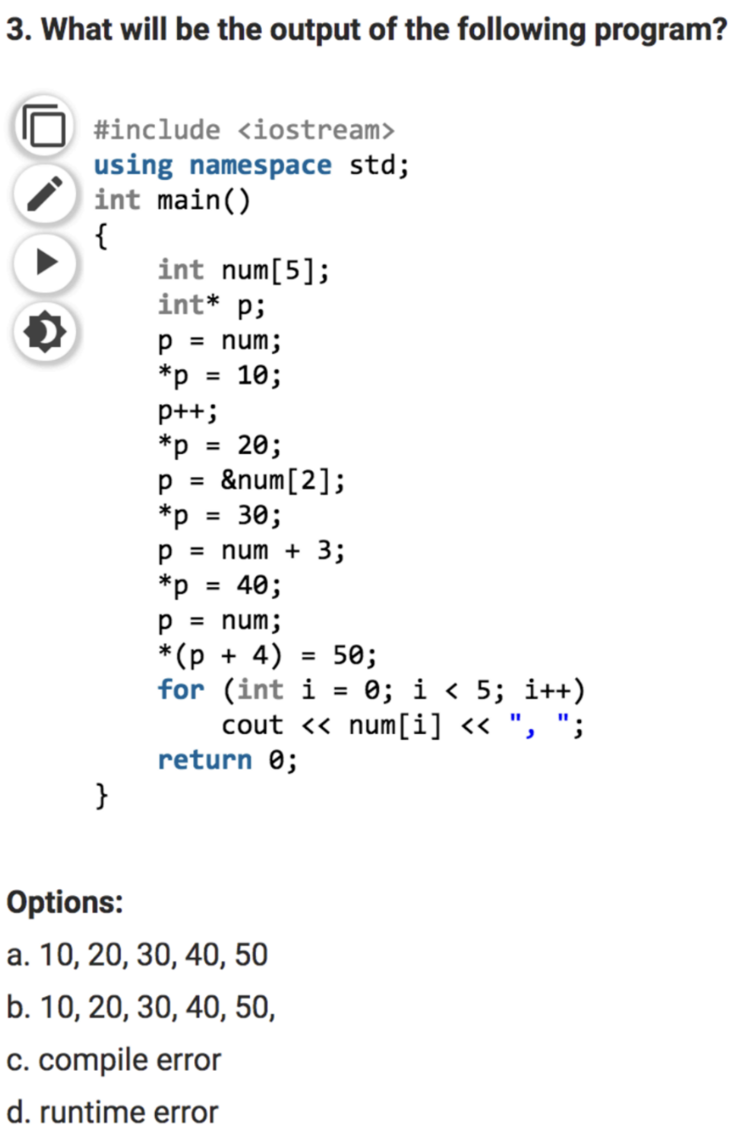
c. 15

d. Compile error

=> d

<u>Explanation</u>: ptr is a constant pointer that points to a constant,means pointer ptr only points to i ,not any other variable, along with that i can only be read and cannot be modified => so in line 3, 4, 5 in main is invalid because these commands are trying to change the value of pointer and i.

Assignment 32



3. What will be the output of the following program?

```
#include <iostream>
using namespace std;
int main()
{
    int num[5];
    int* p;
    p = num;
    *p = 10;
    p++;
    *p = 20;
    p = &num[2];
    *p = 30;
    p = num + 3;
    *p = 40;
    p = num;
    *(p + 4) = 50;
    for (int i = 0; i < 5; i++)
        cout << num[i] << ", ";
    return 0;
}
```

Options:

a. 10, 20, 30, 40, 50

b. 10, 20, 30, 40, 50,

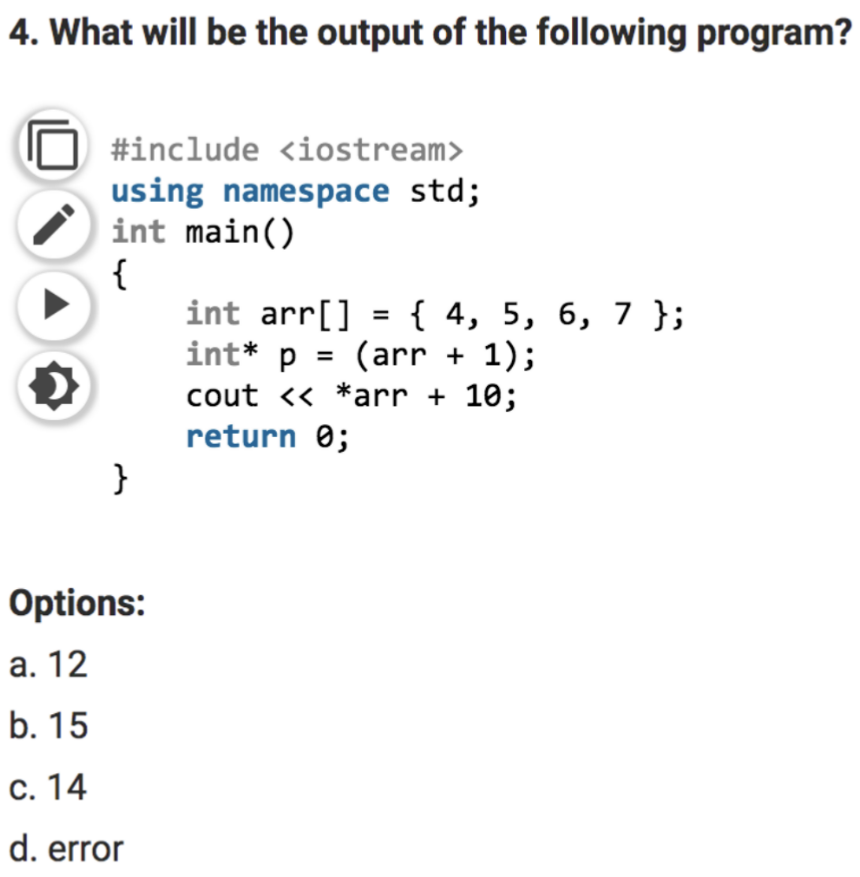
c. compile error

d. runtime error

=> b.

<u>**Explanation**</u>: in main function: line 1 initializes a num array with 5 elements. Lines 2,3 initialize a pointer p and makes it point to the num array (pointing to num[0]). Line 4 gives the value of num[0] equal to 10. Line 5.6 make p point to num[1] and gives the value at there is 20. Lines 7,8 points to num[2] and gives the value there is 30. Line 9,10 points to num[3] and gives the value there 40. Line 9 points to num[0], line 10 gives *(p+4) = 50 means num[4] has value is 50. Use a for loop to output the num array to the screen after making changes to the above values and enclose the "," sign after the last value of the array.

Assignment 33



4. What will be the output of the following program?

```
#include <iostream>
using namespace std;
int main()
{
    int arr[] = { 4, 5, 6, 7 };
    int* p = (arr + 1);
    cout << *arr + 10;
    return 0;
}
```

Options:

a. 12

b. 15

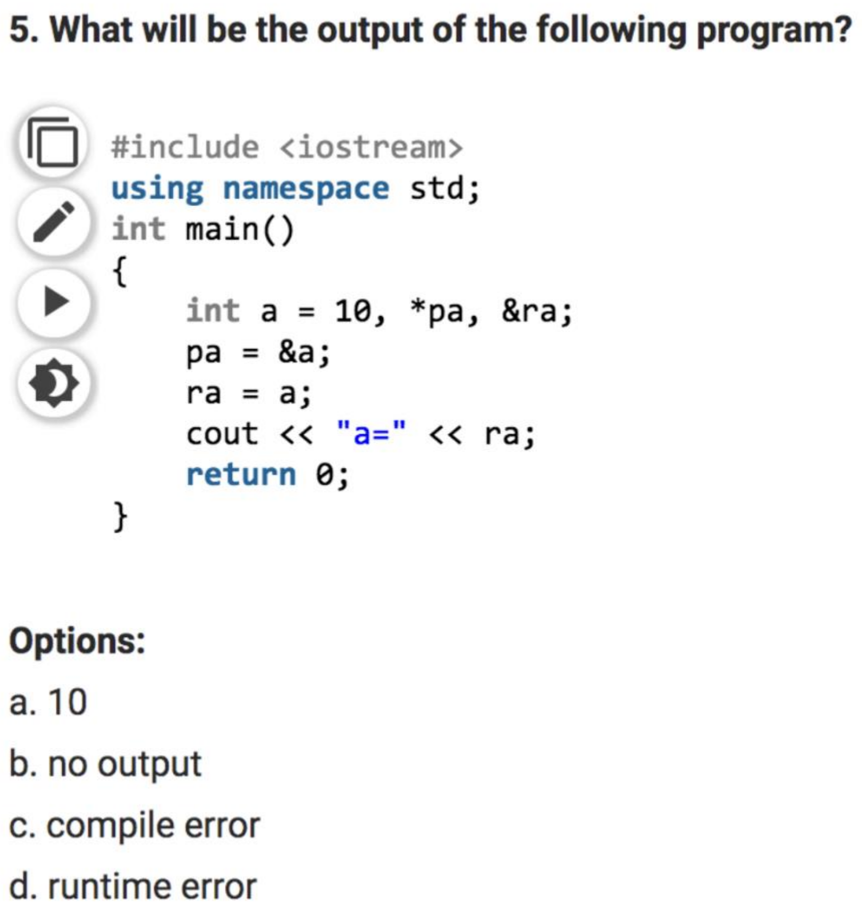
c. 14

d. error

=> c

<u>**Explanation**</u>: an array arr[] is actually a pointer arr pointing to the first element of the array, initialize int *p = (arr+1) ie make p point to the second address of the element of the array. In the command cout << *arr + 10, is to get the value at the first position of the array plus 10, equivalent to the command *&arr[0] + 10, ie take 4 + 10 = 14.

Assignment 34



5. What will be the output of the following program?

```
#include <iostream>
using namespace std;
int main()
{
    int a = 10, *pa, &ra;
    pa = &a;
    ra = a;
    cout << "a=" << ra;
    return 0;
}
```

Options:

a. 10

b. no output

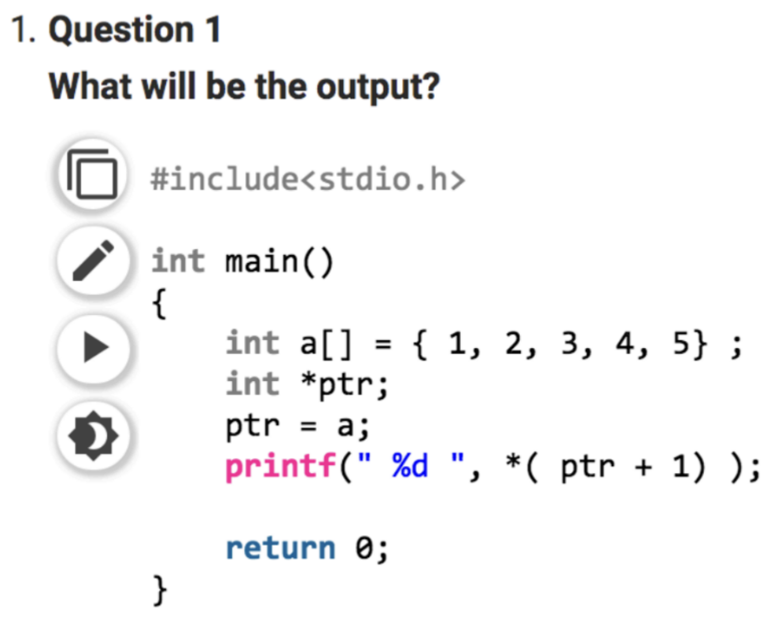
c. compile error

d. runtime error

=> c

Explanation: in line 1 of main function initializes 3 variables: variable a with the value 10, a pointer pa, and a reference variable &ra. In line 2,3, assign a to pointer pa and the reference variable ra => causes a compile error because the pointer can be initialized and assigned it an address at any time, while the reference variable can only be addressed during its initialization, so the ra = a command will cause an error.

Assignment 35



1. **Question 1**

   What will be the output?

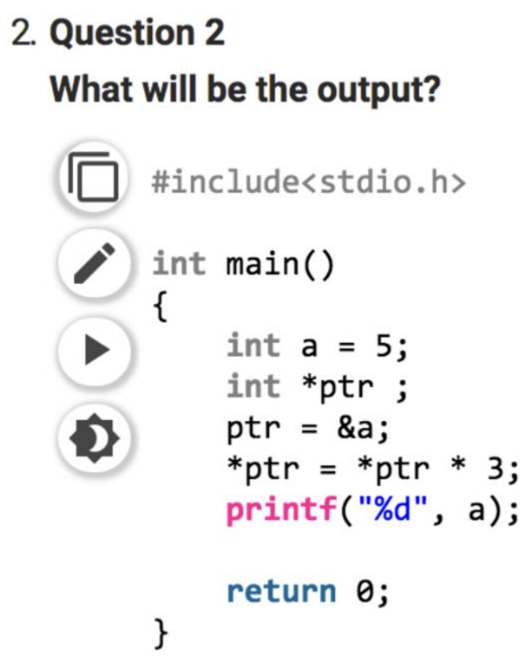
```
#include<stdio.h>

int main()
{
    int a[] = { 1, 2, 3, 4, 5} ;
    int *ptr;
    ptr = a;
    printf(" %d ", *( ptr + 1) );

    return 0;
}
```

=> After running the program,the output is: 2
Explanation: because an array a is assigned to pointer ptr, which means ptr points to the first element of array a which is a[0], then the value at a[0] will be *(ptr+ 0) => command *(ptr+1) means get the value at the 2nd position in the array which is 2.

Assignment 36



2. **Question 2**

What will be the output?

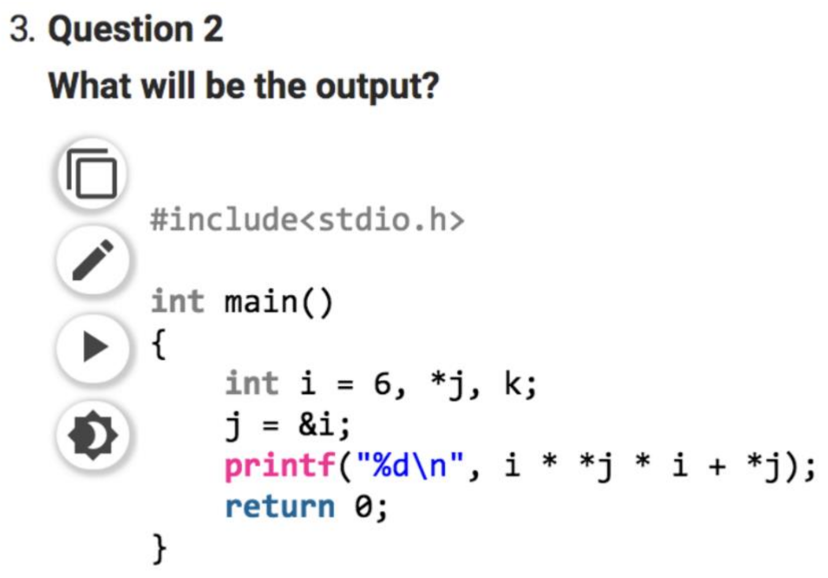
```c
#include<stdio.h>

int main()
{
    int a = 5;
    int *ptr ;
    ptr = &a;
    *ptr = *ptr * 3;
    printf("%d", a);

    return 0;
}
```

=> After running the program,the output is: 15
<u>Explanation</u>: pointer ptr is initialized and points to a,the command “*ptr = *ptr * 3” means get the value at the address that ptr is pointing to and multiplied by 3, so get 5*3 = 15.

Assignment 37



3. **Question 2**

What will be the output?

```
#include<stdio.h>

int main()
{
    int i = 6, *j, k;
    j = &i;
    printf("%d\n", i * *j * i + *j);
    return 0;
}
```

=> After running the program,the output is: 222
Explanation: the 1st line in the main function initializes the variable i with the value 6, an a pointer j and a variable k. The second line assigns pointer j the address of i. In the command "i* *j * j + *j" from left to right means taking i (6) multiplied by the value at the address j points to (the value of i is 6) multiplied by i(6) with the value at the address j points to (6) = > 6 * 6 * 6 + 6 = 222.