50 MCQs ON

“POINTERS AND ARRAY”

PROGRAMMING IN ‘C’

What does the following code print?

```
int x = 10;
int *ptr = &x;
printf("%d", *ptr);
```

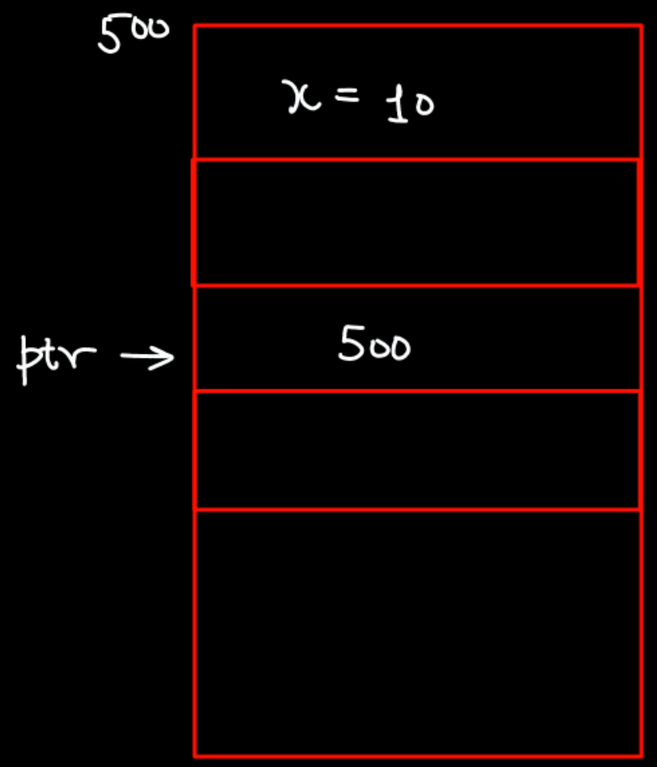
↳ Value at (ptr)
= Value at (500)
= 10

a) Address of x

b) 10

c) Garbage value

d) Compilation error



What is the size of a pointer to an int on a 64-bit system?

a) 2 bytes

b) 4 bytes

c) 8 bytes

d) Depends on the compiler

Which of the following is NOT a valid pointer type?

a) int * → integer

b) void * → void ptr

c) float ** ← double pointer

d) char & ← Invalid

What is a NULL pointer?

a) A pointer to a memory location containing 0

b) A pointer that points to address 0

c) A pointer that stores a garbage value

d) None of the above

```
int *ptr = NULL;
```

$\boxed{ptr = 0}$

What happens if you dereference a NULL pointer?

a) Prints 0

b) Returns NULL

c) Causes a segmentation fault → "An error where a pointer ties to retrieve a value from non-existing memory address/unthorized segment."

taking out

d) Undefined behavior

```
int *ptr = NULL;
```

$\Rightarrow \boxed{ptr = 0}$

```
printf("%d", *ptr);
```

value At (0)

= ?

What will be the output of the following code?

```
int arr[] = {10, 20, 30};
int *p = arr;
printf("%d", *(p + 2));
```

0 1 2

500 504 508

ptr = arr = &arr[0]

p = 500

ValueAt(p+2)

Value At(500+2)

Value At(500+2*4)

Value At(508)

=30

ptr + int => ptr + int * sizeof(arr[0])

a) 10

b) 20

✓ c) 30

d) Compilation error

What is the difference between const int *p and int *const p?

a) Both are the same

b) const int *p prevents modifying the value being pointed to, int *const p prevents changing the pointer itself

c) const int *p prevents modifying the pointer, int *const p prevents modifying the value being pointed to

d) None of the above

Const ← Make the value read only.

m = 265;

Const int *p = &m;

Can not change the value at &m.

int *Const p = &m;

Make the point Constant → This pointer will not change the address.

Constant pointer

What does the following code snippet do?

```
int x = 5;
int *p = &x;
*p = 10;
printf("%d", x);
```

x = ~~5~~ 10
p = &x

10

Value at (p) = 10

a) Prints 5

b) Prints 10

c) Compilation error

d) Undefined behavior

What does the following expression compute?

```
int *p;
p + 1;
```

ptr + int ⇒ ptr + i x sizeof(datatype)

a) Moves p to the next memory block of the same data type

b) Adds 1 byte to p

c) Causes undefined behavior

d) None of the above

Which of the following statements is correct?

a) A pointer must always point to a valid memory location

*p = NULL

b) A pointer can point to any data type

c) A pointer cannot point to a function

d) Pointer arithmetic can only be performed on int *

What is the value of *(arr + 1) if arr is an array defined as int arr[] = {3, 6, 9};?

500 504 508

a) 3

b) 6

c) 9

d) Undefined behavior

int arr[] = {3, 6, 9}

*(arr + 1)

⇒ Value At (arr + 1)

= &arr[0] + 1

= 6

arr = 500

⇒ *(500 + 1)

*(500 + 1×4)

500 + 4

*(504)

What happens if you dereference an uninitialized pointer?

a) Undefined behavior

b) Prints a garbage value

c) Compilation error

d) Returns NULL

int *ptr; ← uninit

Which operator is used to access the value stored at the address a pointer points to?

a) * ← dereferentiation operator ← value at( )

b) &

c) ->

d) .

What does the following code print?

```
int x = 5, y = 10;
int *p = &x, *q = &y;
printf("%d", *p + *q);
```

a) 5

b) 10

c) 15

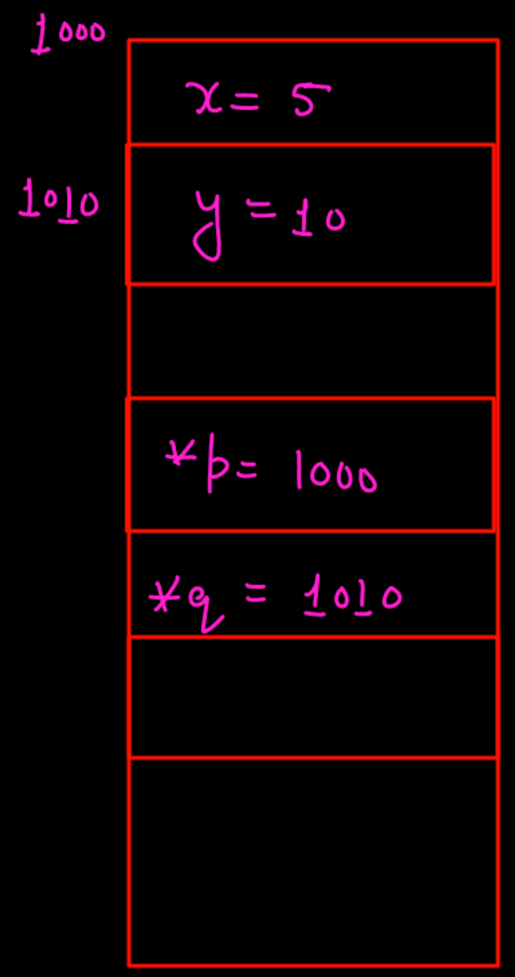
d) Compilation error

Value at (p) + Value at (q)

Value at (1000) + Value at (1010)

5 + 10

= 15



What is a dangling pointer?

a) A pointer initialized to 0

b) A pointer pointing to deallocated memory

c) A pointer pointing to NULL

d) A pointer to an uninitialized variable

Which of the following is equivalent to *(arr + i) in an array?

a) arr[i]

b) &arr[i]

c) arr + i

d) *(arr[i])

arr = & arr[0]

* (arr + i)

Value at (arr + i)

⇒ Value at (Base add + offset * sizeof(arr[0]))

⇒ Value at (&arr[0] + i * sizeof(arr[0]))

Value at (& arr[i])

⇒ ~~* &~~ arr[i]

= arr[i]

What is the output of this code?

```
int x = 10, y = 20;
int *p = &x;
*p = *p + y;
printf("%d", x);
```

a) 10

b) 20

c) 30

d) Compilation error

$x = \cancel{10}\ 30$

$y = 20$

$*p = &x$

$*p = *p + y$

$\Rightarrow$ Value at (p) = Value at $(p) + y$

Value at $(p) = 10 + 20$

Value at $(p) = 30$

Value at $(100) = 30$

$&x = 100$ ← Assume

$p = 100$

How do you declare a constant pointer to an integer?

a) int const *p;

b) int *const p;

c) const int *p;

d) Both b and c

What is the output of the following code?

```
int arr[3] = {1, 2, 3};
int *p = arr;
printf("%d", *(++p));
```

100 104 108

arr = 100

p = 100

Value at (p+1)

Value at (100+1*4)

*(104)

⇒ 2

a) 1

b) 2

c) 3

d) Compilation error

Can a pointer be initialized to NULL?

a) Yes

b) No

Which statement about arrays and pointers is correct?

a) Arrays and pointers are the same in C ✗

b) The name of an array is a pointer to its first element ✓

c) Arrays are passed to functions by value ✗

d) None of the above ✗

arr = &arr[0]

What does the following code output?

```
int x = 10;
int *p = &x;
printf("%p", p);
```

a) Address of x

b) Value of x

c) Garbage value

d) Compilation error

Which of the following is true about void * pointers?

a) They can store addresses of any data type

b) They can be dereferenced directly

c) They are not valid in C

d) They are equivalent to NULL pointers

integer ptr.

⇒ int * ptr

```
int x = 20;
Void *ptr = &x
printf("%d", *ptr);  ← Error
```

⇒ Void pointers can not be directly dereferentiate

⇒ printf("%d", *(int*)(ptr));

What will the following code output?

100 104 108 112

arr = 100

```
int arr[] = {1, 2, 3, 4};
int *p = arr + 2;
printf("%d", *p);
```

p = 100+(2 * 4)

p = 100+8 = 108

Value at (p)

Value at (108)

= 3

a) 1

b) 2

c) 3

d) 4

What is the output of this code?

```
int x = 100;
int *p = &x;
printf("%p", (void *)p);
```

a) Address of p

b) Address of x

c) Value of x

d) Compilation error

add(x) = 500

x = 100

*p = &x → p = 500

"typecasting the pointer from int to void"

print करो → value at (p) लेकिन किसी को मत बताओ कि p एक int ptr है।

Which of the following is an invalid declaration?

a) int *p; → single ptr

b) float **p; → double ptr

c) char ***p; → pointer to pointer

d) void p;

What happens when a pointer is incremented?

a) It moves to the next byte in memory

b) It moves to the next memory location of its data type size

c) Undefined behavior

d) Compilation error

ptr + 2

= ptr + 2 * sizeof (datatype)

What does the & operator do when used with a variable?

a) It dereferences the variable

b) It provides the address of the variable

c) It declares a pointer

d) It increments the variable

Which of the following operations is invalid on pointers?

a) Addition of an integer → ptr + int ✓

b) Subtraction of an integer → ptr - int ✓

c) Comparison of two pointers → ptr1 == NULL, ptr1 == ptr2 ✓

d) Multiplication of a pointer by an integer ✗

Not allowed

ptr * 3 ← Error

✓ ptr + i * sizeof(datatype)

✓ ptr - 1 * (sizeof(datatype))

What does the following code output?

arr = 100

p = 100

100 104 108

```
int arr[] = {5, 10, 15};
int *p = arr;
printf("%d", *(p + 1) + *(p + 2));
```

Value at (100+1) + Value at (100+2)

Value at (104) + Value at (108)

10 + 15

= 25

a) 15

b) 20

c) 25

d) Compilation error

What is the output of this code?

```
int x = 10, y = 20;
int *p = &x, *q = &y;
printf("%d", *p - *q);
```

$x = 10$ $p = &x$

$y = 20$ $q = &y$

Value at (p) − Value at (q)

$10 - 20$

$= -10$

a) -10

b) 10

c) 30

d) Compilation error

How do you declare a pointer to a function that returns an integer?

a) int func*();

b) int (*func)();

c) int *func();

d) int **func();

```
void fun (int x){
    printf("%d", x);
}
int main(){
    void (*fptr)(int) = &fun;
    (*fptr)(25);
    return 0;
}
```

function pointer

Calling a function using pointer

What will the following code output?

```
int arr[] = {10, 20, 30};
int *p = arr + 1;
printf("%d", *(p - 1));
```

100 104 108

arr = 100

p = arr + 1
= 100 + 1 x 4
= 104

Value at (p - 1)
(104 - 1 x 4)
value at (100)
= 10

a) 10

b) 20

c) 30

d) Compilation error

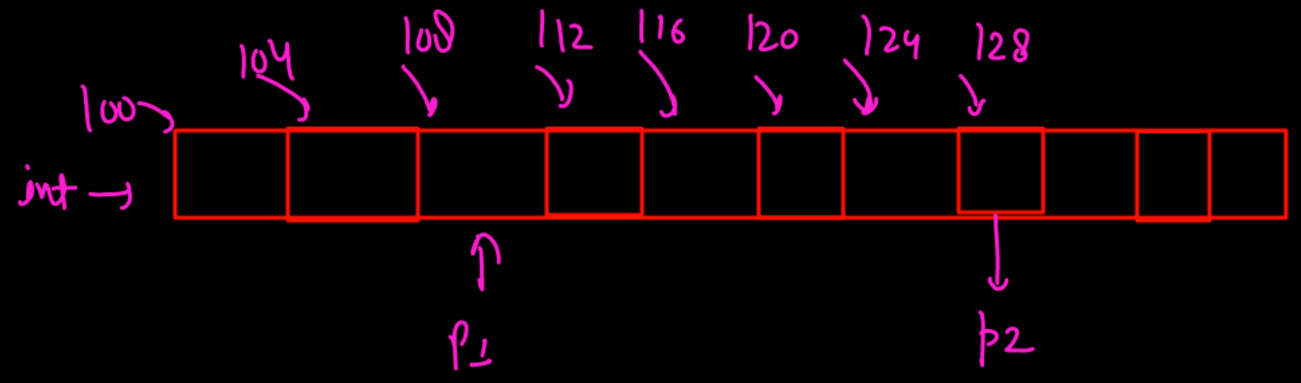
What is the result of p2 - p1 if p1 and p2 are pointers to elements of the same array?

a) The difference in bytes

b) The difference in the number of elements between the pointers

c) Undefined behavior

d) Compilation error



$$p2 - p1$$
$$= 128 - 108$$
$$= 20 \text{ Bytes} / 4$$
$$= \boxed{5 \text{ elements}}$$

What is the output of this code?

```
int x = 5;
int *p = &x;
printf("%d", *p++);
```

5

*p++ ← postfix increment

~~*~~ &x

= x

a) 5

b) Undefined behavior

c) Address of x

d) Compilation error

What will the following code output?

100 104 108

```
int arr[3] = {1, 2, 3};
int *p = arr;
printf("%d", *(p + 3));
```

p = 100

*(p+3)

Value at (100+3x4)

Value at (100+12)

Value at (112)

a) 3

b) Garbage value

c) Undefined behavior ⇒

d) Compilation error

Which of the following is a valid pointer arithmetic operation?

a) p1 + p2 ✗

b) p1 - p2 ✓

c) p1 * 2 ✗

d) p1 / 2 ✗

What happens when you pass a pointer to a function?

Call by Reference

actual Address

a) A copy of the pointer is passed

b) The function receives the actual address stored in the pointer

c) The function receives the value at the pointer's address

d) None of the above

Which of the following is not a use of pointers?

a) Accessing array elements

b) Dynamic memory allocation

c) Passing arrays to functions

d) Returning multiple values from a function

Not allowed

What will be the output of this code?

```
int x = 5;
int *p = &x;
int **pp = &p;
printf("%d", **pp);
```

a) Address of x

b) 5

c) Address of p

d) Compilation error

$x = 5$

$*p = &x$

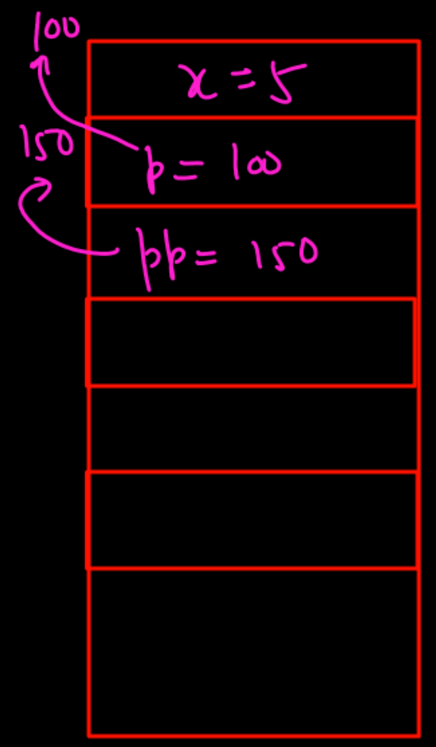
$**pp = &p$

Value at (Value at (pp))

Value at (Value at (150))

Value at (100)

$= 5$



Which of the following statements about pointers is FALSE?

a) Pointers can be incremented and decremented

b) Pointers can be compared

c) Pointers can be multiplied

d) Pointers can point to other pointers

↳ pointer to pointers

How do you check if a pointer is valid?

a) Compare it with NULL

b) Dereference it and check its value

c) Check if it contains 0

d) None of the above

```
int x = 20;
int *ptr = &x
if ptr==NULL {
    print("Invalid")
```

What is the output of this code?

```
int x = 10;
int *p = &x;
*p = *p + 5;
printf("%d", x);
```

15

Value at (p) = Value at (p) + 5
= 10 + 5
= 15

a) 10

b) 15

c) Undefined behavior

d) Compilation error

Which of the following is an invalid pointer assignment?

a) int *p = NULL;

b) int *p = &x;

c) int *p = 1000; ← Invalid

d) int *p = arr;

What is the size of a pointer to a void on a 64-bit system?

a) 2 bytes

b) 4 bytes

c) 8 bytes

d) Depends on the data type

Which of the following is NOT a feature of pointers?

a) Indirect access to variables

b) Dynamic allocation

c) Pointer arithmetic

d) Implicit type conversion → float → int

What is the output of this code?

```
int x = 5;
int y = 10;
int *p = &x, *q = &y;
p = q;
printf("%d", *p);
```

$p = q \rightarrow p = 200 = q$

a) 5

b) 10

c) Address of y

d) Compilation error

$p = \&x = 100$

$q = \&y = 200$

$x = 5, \; y = 10$

$p = q$

$p = 200$

What does const int *p mean?

a) p is a constant pointer

b) The value pointed to by p cannot be changed

c) Both p and the value cannot be changed

d) None of the above

int x = 20

Const int *p = &x

The Value of x will not change

What will the following code output?

```
int arr[] = {1, 2, 3};
int *p = arr;
*p++ = 10;
printf("%d", arr[0]);
```

100 104 108

10

p = 104

100

Value at (p++) = 10

Value at (100) = 10

10

a) 1

b) 10

c) Garbage value

d) Compilation error

What happens if you dereference a void * pointer?

a) Compilation error

b) Prints 0

c) Undefined behavior

d) Requires typecasting to dereference

void *ptr;

ptr = &x;

ptr → 8 bytes

What is the output of this code?

$*p = \&x = 100$

```
int x = 10;
int *p = &x;
printf("%d", *(p + 1));
```

Value at (p+1)

= 100 + 1 * sizeof(int)

=(104)

Value at (104)

a) 10

b) Garbage value

c) Undefined behavior

d) Compilation error