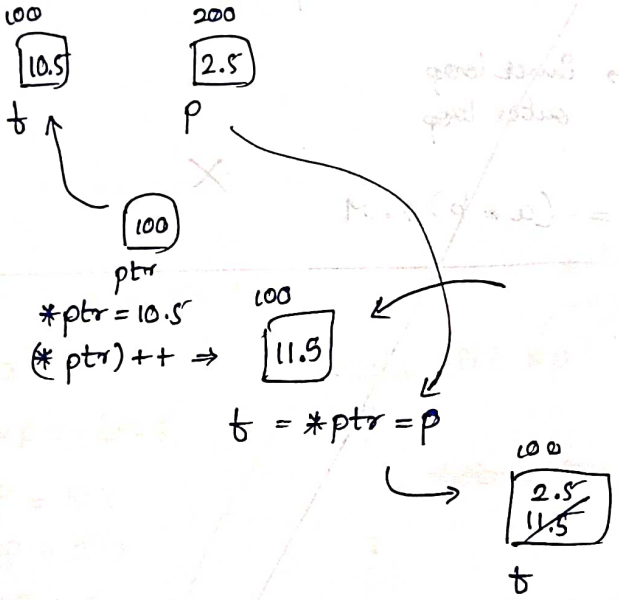
POINTERS PROBLEMS

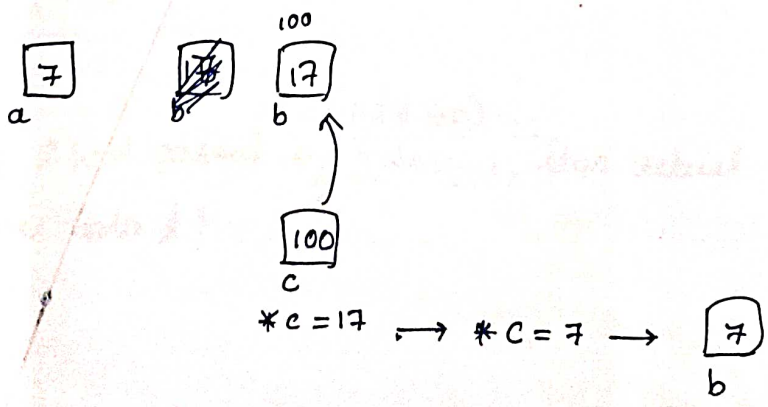
Q.1)
```
float f = 10.5;
float p = 2.5;
float *ptr = &f;
(*ptr)++;
*ptr = p;
cout << *ptr << " " << f << " " << p;
```

Sol:



∴ O/p : 2.5 2.5 2.5

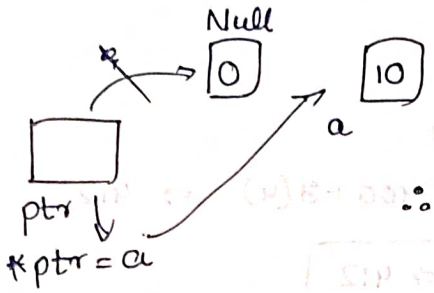
Q.2)
```
int a = 7; int b = 17; int *c = &b; *c = 7;
cout << a << " " << b << endl;
```



O/p:- 7 7

Q:
```
int *ptr = 0;
int a = 10;
*ptr = a;
cout << *ptr << endl;
```
→ dereferencing a null pointer gives Runtime ERROR

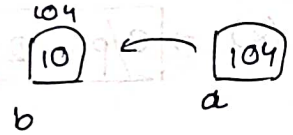
Sol:



∴ O/p = ~~10~~ Runtime error.

Q: Which of the following gives the memory adress of the variable 'b' pointed by pointer 'a' i.e
```
int b = 10;
int *a = &b;
```

Sol:



we need memory adress of b

∴ ans = a ✓

cout << a << endl;

Q: What will be output?
```
char ch = 'a';
char *ptr = &ch;
ch++;
cout << *ptr << endl;
```

Sol:

ch++ ⇒ a++ → b

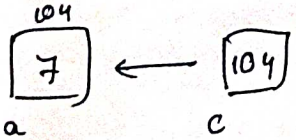
ch.

a = ~~65~~ 97
+1
98 = b

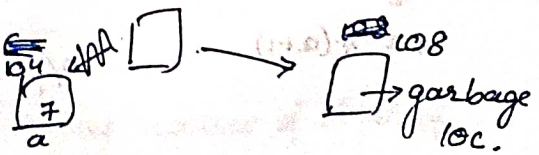
∴ *ptr = b ✓

Q: `int a = 7; int *c = &a; c = c+1; cout << *a << " " << *c << endl;`

Sol:



c = c+1
= 104 + 1(4)
= 108.

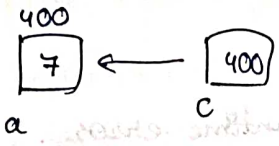


O/p ⇒ 7 garbage value.

Q: memory address of a = 400.

```
int a = 7;
int *c = &a;
c = c + 3;
cout << c << endl;
```

Sol:



c = 400

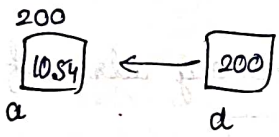
c + 3 ⇒ 400 + 3(4) ⇒ 412

o/p ⇒ 412

Q: address of a = 200

```
double a = 10.54; double *d = &a; d = d + 1;
cout << d << endl;
```

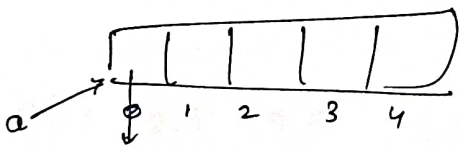
Sol:



d = 200 + 1(8) ⇒ o/p = 208

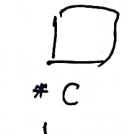
Q: `int a[5]; int *c; cout<<sizeof(a) << " " << sizeof(c);`

Sol:



int = 4

size = 4 × 5 = 20.



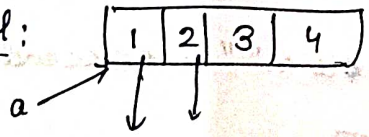
size = 8.

o/p: 20 8

Q: `int a[] = {1, 2, 3, 4}`

`cout << *(a) << " " << *(a+1);`

Sol:

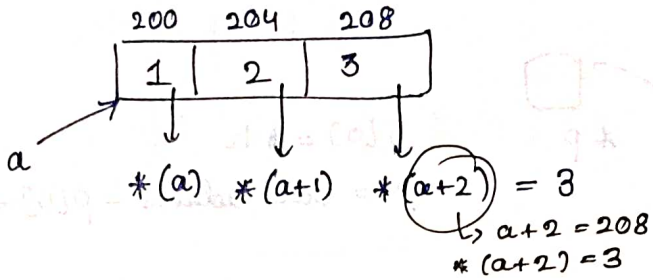


o/p: 1 2

Q: Adress of 0th index of array 'a' is 200.

```
int a[3] = {1,2,3};
cout << *(a+2);
```

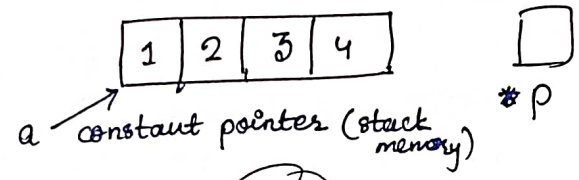
Sol:



o/p : 3

Q:
```
int a[] = {1,2,3,4};
int *p = a++;
cout << *p << endl;
```

Sol:

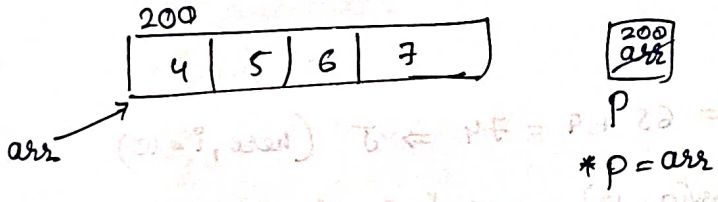


but *p = a++ → *p = (a = a+1) → Not possible

o/p : Runtime ERROR

Q:
```
int arr[] = {4,5,6,7};
int *p = (arr+1);
cout << *arr + 9;
```

Sol:



*arr + 9

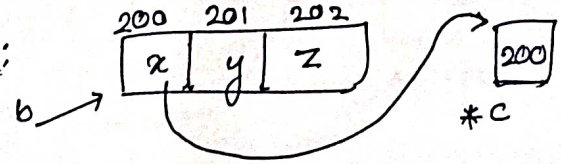
4 + 9 ⇒ 13

o/p ⇒ 13

Q: adress of 0th index of array is 200.

```
char b[] = "xyz";
char *c = &b[0];
cout << c << endl;
```

Sol:

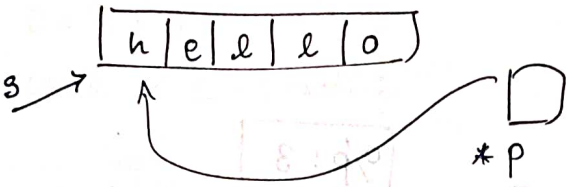


c stores adress of start of array b → char array. cout<<b ⇒ xyz

∴ cout << c → xyz

Q) char s[] = "hello";
char *p = s;
cout << s[0] << " " << p[0];

Sol:

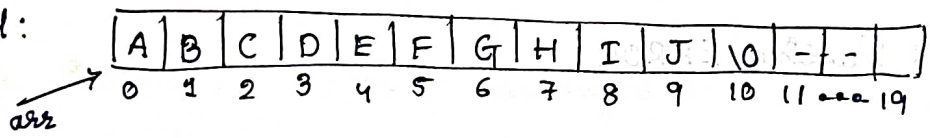


s[0] = h
p = base address = p[0] = h

o/p : h h

Q) char arr[200];
int i;
for(i=0; i<10; i++)
{
*(arr+i) = 65+i;
}
*(arr+i) = '\0';
cout << arr;

Sol:

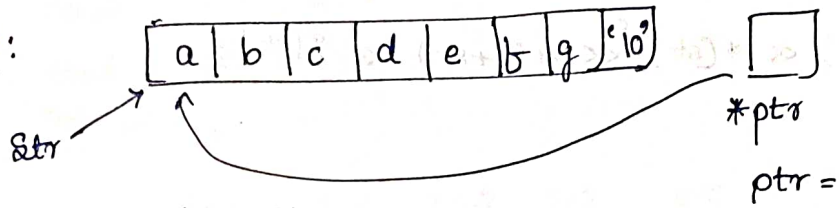


i=0 → *(arr+i) ⇒ 0+0 ⇒ arr[0] = 65+0 = 65 = A (because char arr)
i=1 → arr[1] = 65+1 = 66 = B
i=2 → arr[0+2] = 65+2 = 67 = C
⋮
i=9 → arr[9] = 65+9 = 74 ⇒ J (here, i=10)
*(arr+i) = '\0' → arr[0+10] → arr[10] = '\0'
∴ cout << arr

↳ ABCDEFGHIJ

Q:
```
char *ptr;
char str[] = "abcdefg";
ptr = str;
ptr += 5;
cout << ptr;
```

Sol:



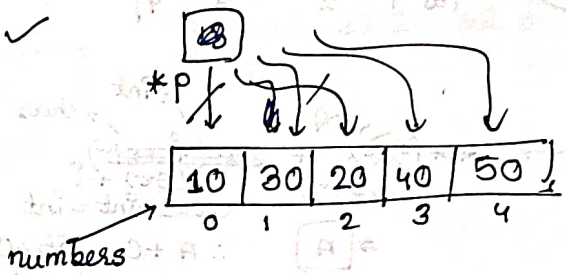
ptr = a + 5
= 97 + 5 ⇒ 102 = f.

cout << str
↳ abcdefg.

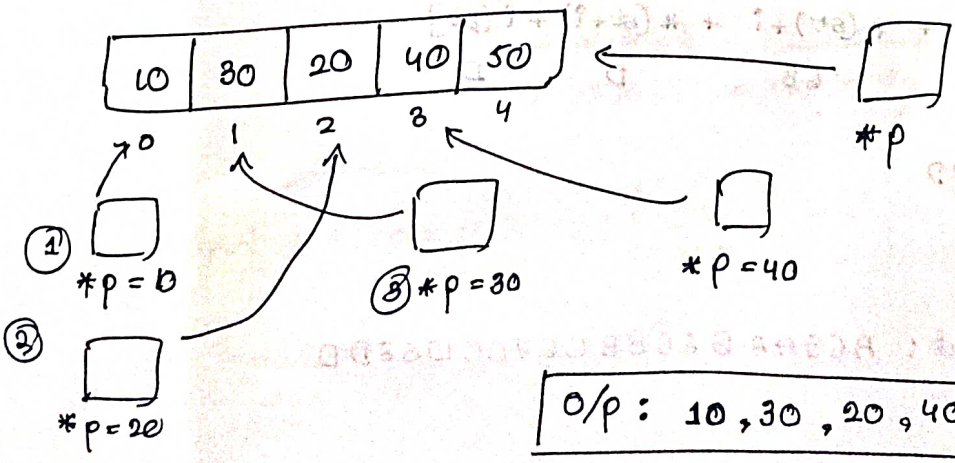
ptr is at f.

cout << ptr ⟹ fg ✓

Q:
```
int main()
{
    int numbers[5];
    int *p;
    p = numbers;
    *p = 10;
    p = &numbers[2];
    *p = 20;
    p--;
    *p = 30;
    p = numbers + 3;
    *p = 40;
    p = numbers;
    *(p+4) = 50;
    for (int n=0; n<5; n++)
    {
        cout << numbers[n] << ",";
    }
    return 0;
}
```



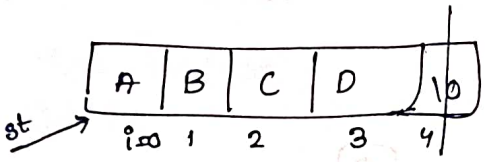
output
⇒ 10, 30, 20, 40, 50



O/p: 10, 30, 20, 40, 50 ✓

```
#include <iostream>
using namespace std;
int main()
{
    char st[] = "ABCD";
    for (int i=0; st[i] != '\0'; i++)
    {
        cout << st[i] << *(st)+i << *(i+st) << i[st];
    }
    return 0;
}
```

Sol:



i=0 to i=3 ✗

i=0 → AAA_BAC_CAD_DAA_

i=0 → A + *(st) + *(i+st) + i[st]

st[i] ⇒ A

*(st)+i : int + char = int; ∴ A + 0 = 65 + 0 = 65

*(0+st) = *(st) = A

0[st] = st[0] = A

o/p = A65AA

i=1 → st[i] + *(st)+i + *(i+st) + i[st]

= B ; A + 1 ⇒ 65 + 1 = 66 ; *(1+st) ⇒ B ; 1[st] = st[i] = B

o/p = B66BB

i=2 → st[i] + *(st)+i + *(st+i) + i[st]

C ; A + 2 ⇒ 67 ; ⇒ C ; ⇒ C

o/p = C67CC

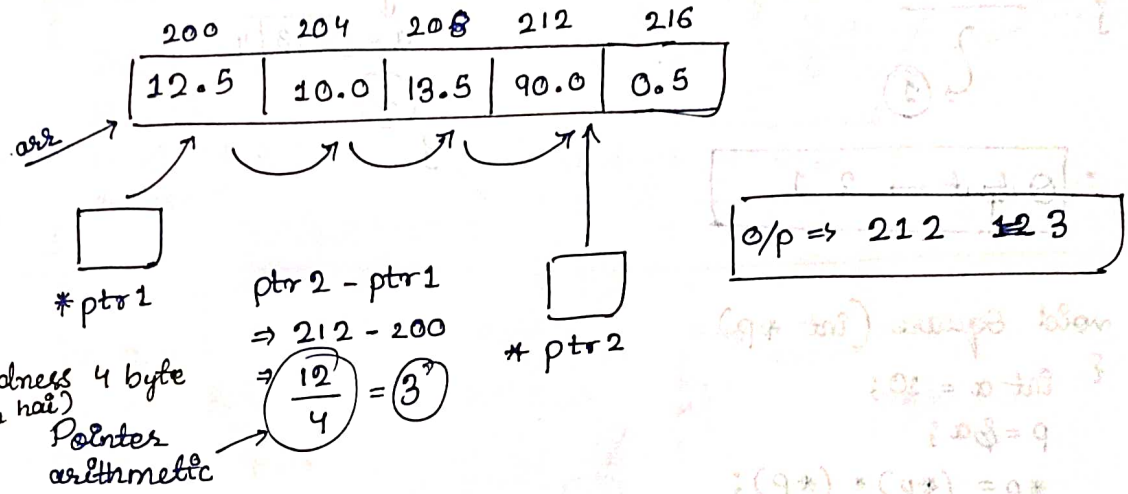
i=3 → st[i] + *(st)+i + *(st+i) + i[st]

D ; 68 ; D ; D

o/p = D68DD

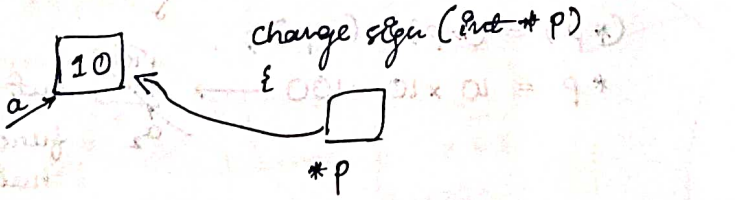
∴ Output: A65AAB66BBC67CCD68DD

```
int main()
{
    float arr[5] = {12.5, 10.0, 13.5, 90.0, 0.5};
    float *ptr1 = &arr[0];
    float *ptr2 = ptr1 + 3;
    cout << ptr2 << " ";
    cout << ptr2 - ptr1;
    return 0;
}
```



Q>
```
void changeSign (int *p)
{
    *p = (*p) * -1;
}
int main()
{
    int a = 10;
    changeSign (&a);
    cout << a << endl;
}
```

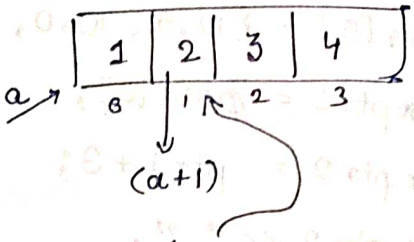
Sol:



*p = *p * -1
= a * -1
⇒ 10 * -1 ⇒ a = -10

Output → cout << a → -10 ✓

Q1
```
void fun (int a[])
{
    cout << a[0] << " ";
}
int main()
{
    int a[] = {1,2,3,4};
    fun (a+1);
    cout << a[0];
}
```



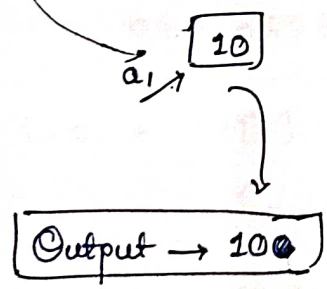

```
fun (a[])
{
    cout<<a[0]<<" ";
}
```

a = 2 3 4 → 2

(1) (2)

∴ Output → 2 1

Q2
```
void Square (int *p)
{
    int a = 10;
    p = &a;
    *p = (*p) * (*p);
}
int main()
{
    int a = 10;
    Square (&a);
    cout << a << endl;
}
```



Square (&a = *p)
{
a2 → 10

(*p) = (*p) * (*p);

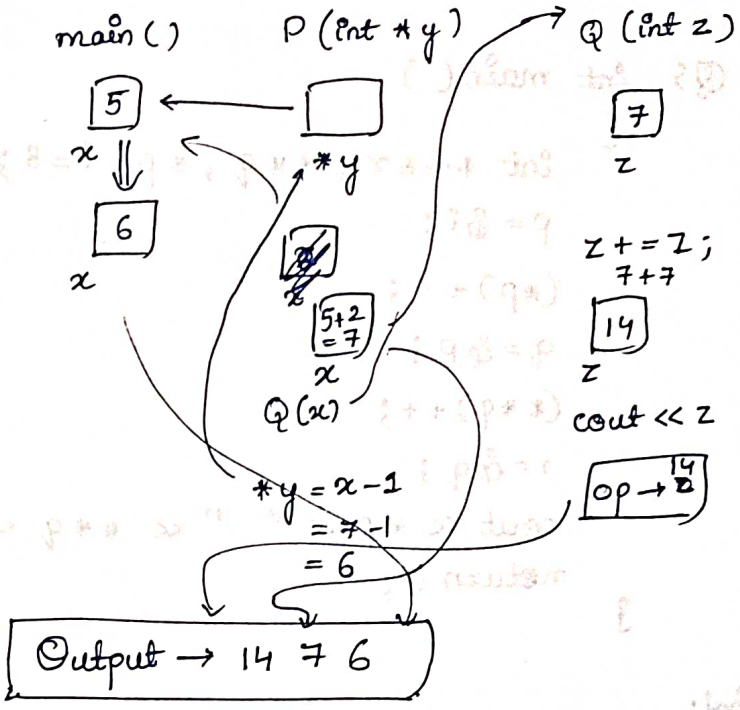
*p = 10 × 10 = 100 → 100

a2

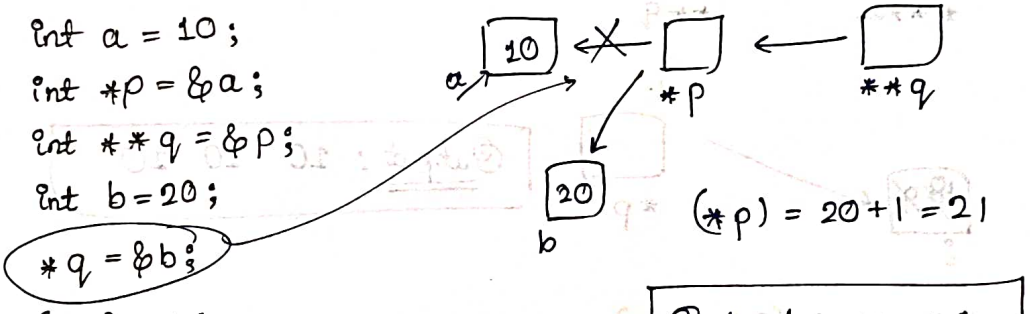
but yo iss function ka a hai. So, 10 (a1)

Q3

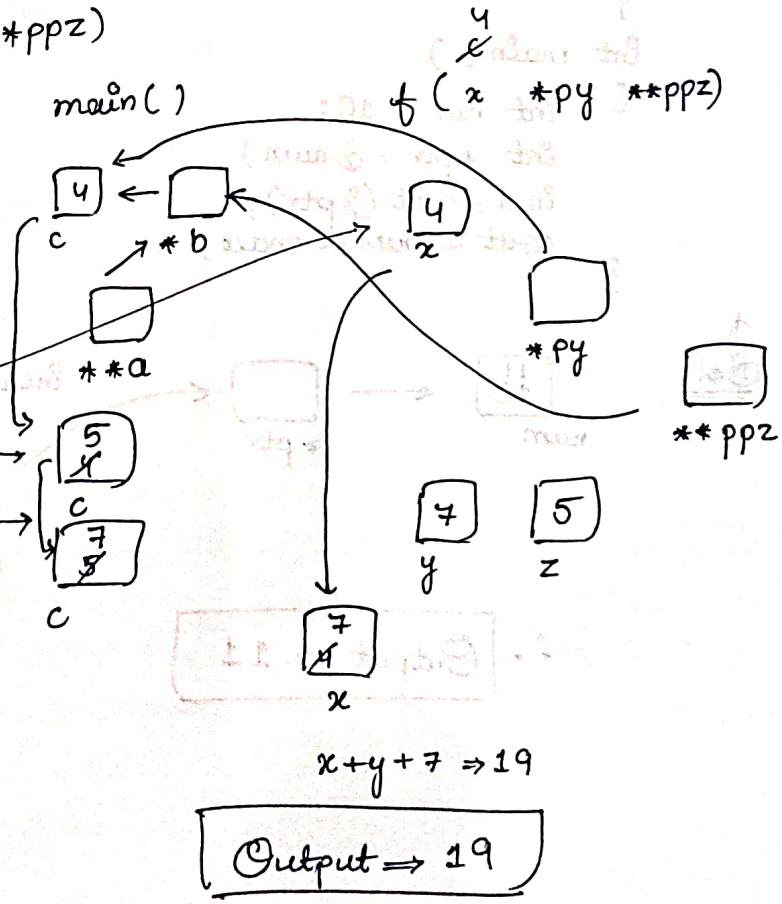
Q;
```
void Q(int z)
{
    z += z;
    cout << z << " ";
}
void P(int *y)
{
    int x = *y + 2;
    Q(x);
    *y = x - 1;
    cout << x << " ";
}
int main()
{
    int x = 5;
    P(&x);
    cout << x;
    return 0;
}
```



Q;
```
int a = 10;
int *p = &a;
int **q = &p;
int b = 20;
*q = &b;
(*p)++;
cout << a << " " << b << endl;
```

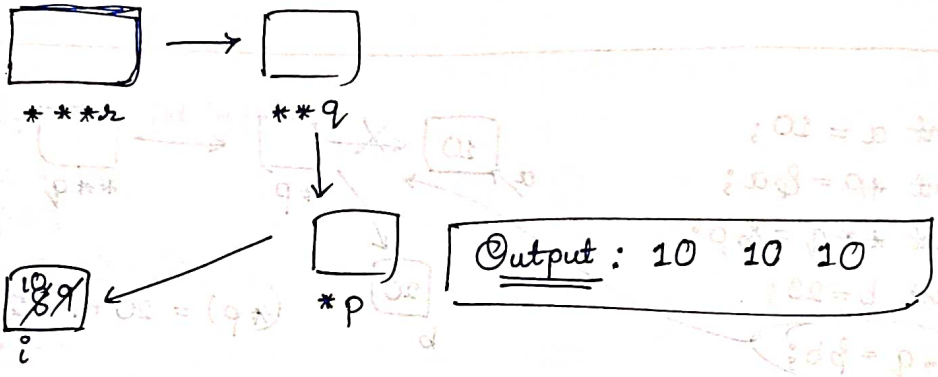


Q;
```
int f(int x, int *py, int **ppz)
{
    int y, z;
    *** ppz += 1;
    z = ** ppz;
    *py += 2;
    y = *py;
    x += 3;
    return x+y+z;
}
int main()
{
    int c, *b, **a;
    c = 4;
    b = &c;
    a = &b;
    cout << f(c, b, a);
    return 0;
}
```



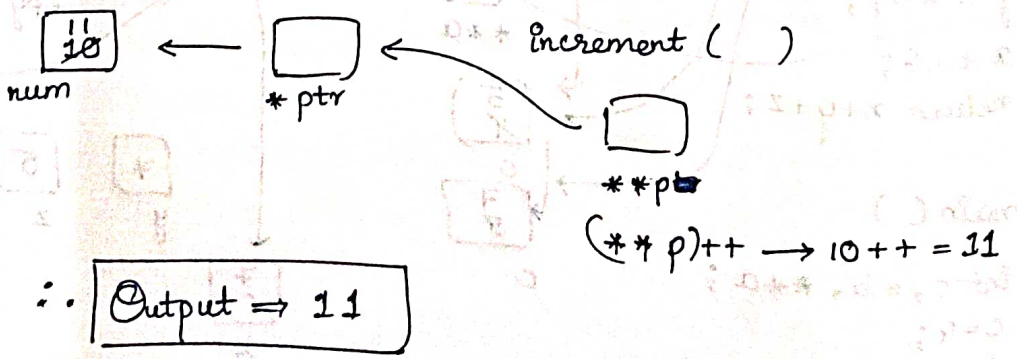
Q)
```
int main()
{
    int ***r, **q, *p, i=8;
    p = &i;
    (*p)++;
    q = &p;
    (**q)++;
    r = &q;
    cout << *p << " " << **q << " " << ***r;
    return 0;
}
```

Sol:



Q)
```
void increment (int **p)
{
    (**p)++;
}
int main()
{
    int num = 10;
    int *ptr = &num;
    increment (&ptr);
    cout << num << endl;
}
```

Sol:



∴ Output ⇒ 11