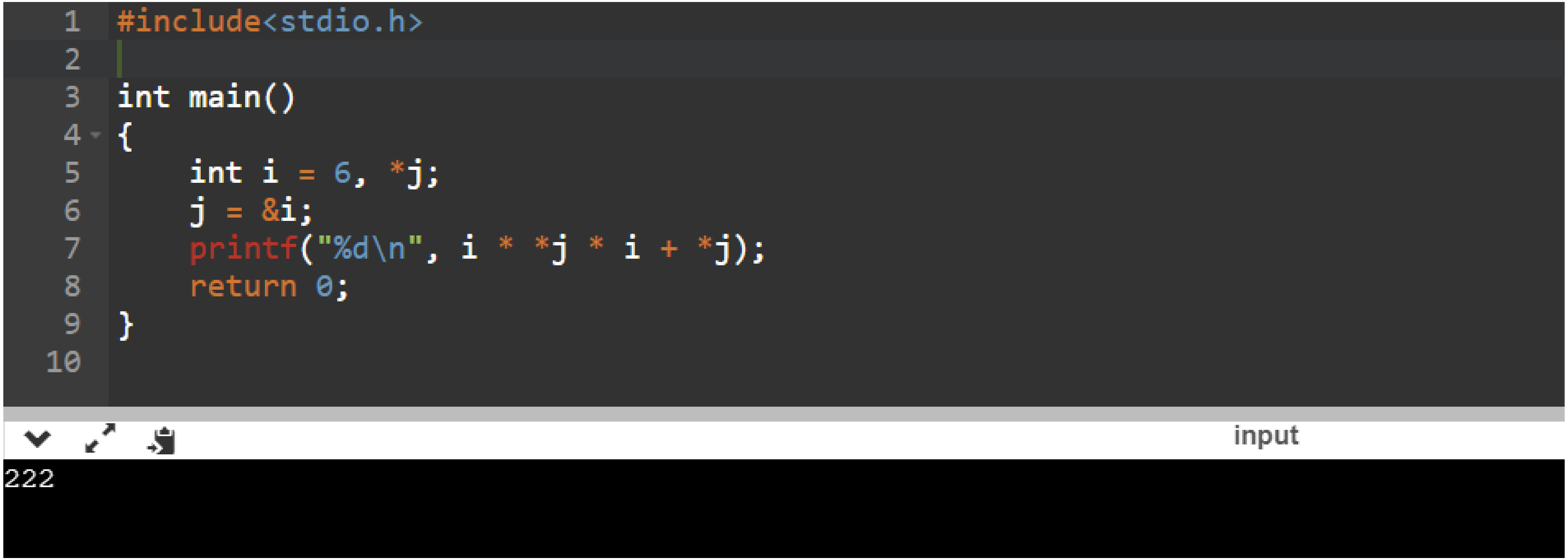
CIS 2107 Midterm Review

Hints

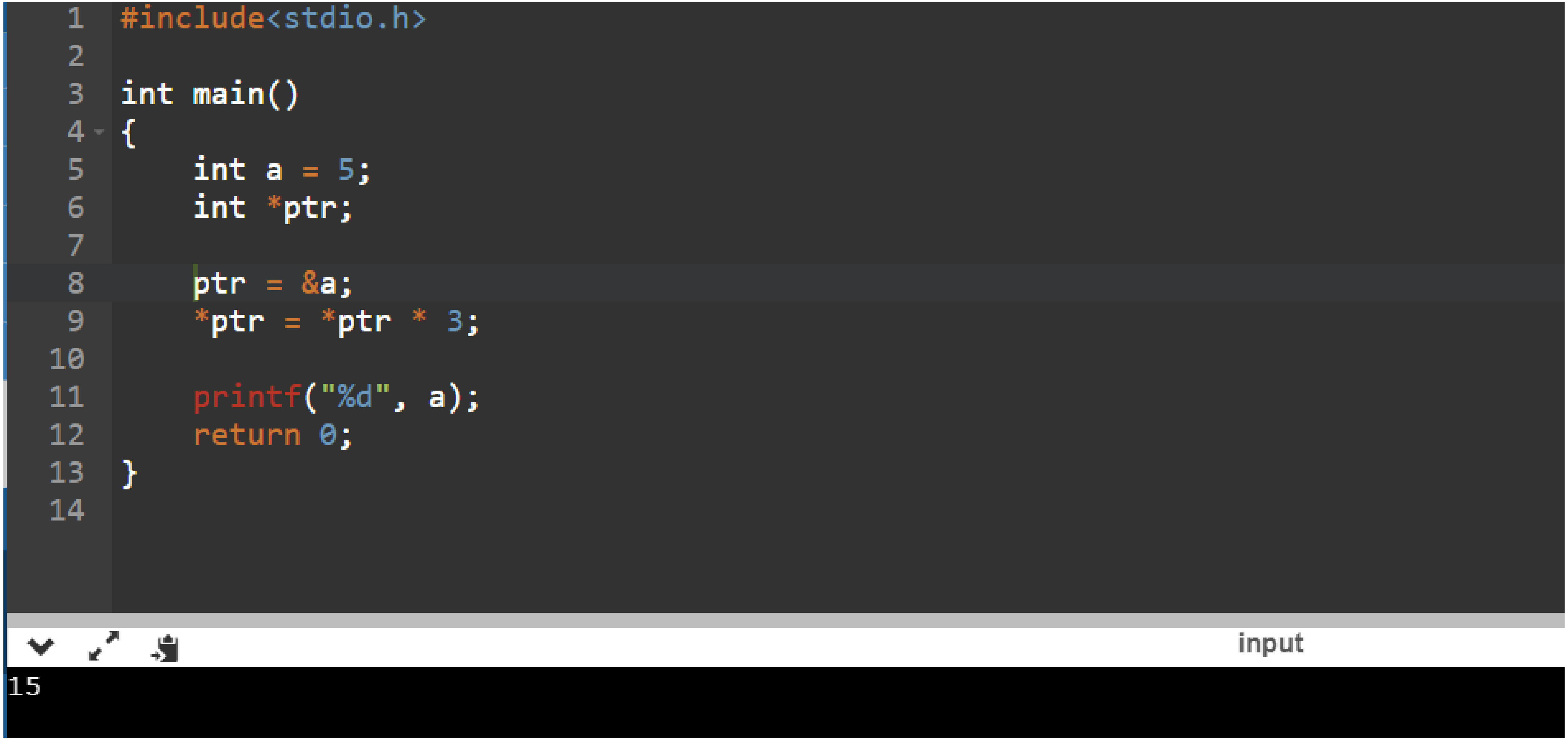
- Your goal all the time like we did in class is to draw memory boxes and set up connections between data and pointers.

- For reading pointers declarations, remember the golden rule **RTL (Right to Left)** : if a pointer confuses you ,then read it from right to left, and as you travel on that direction, say it as you see it.



```c
#include<stdio.h>

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

```

```
222
```



```c
#include<stdio.h>

int main()
{
    int a = 5;
    int *ptr;

    ptr = &a;
    *ptr = *ptr * 3;

    printf("%d", a);
    return 0;
}

```

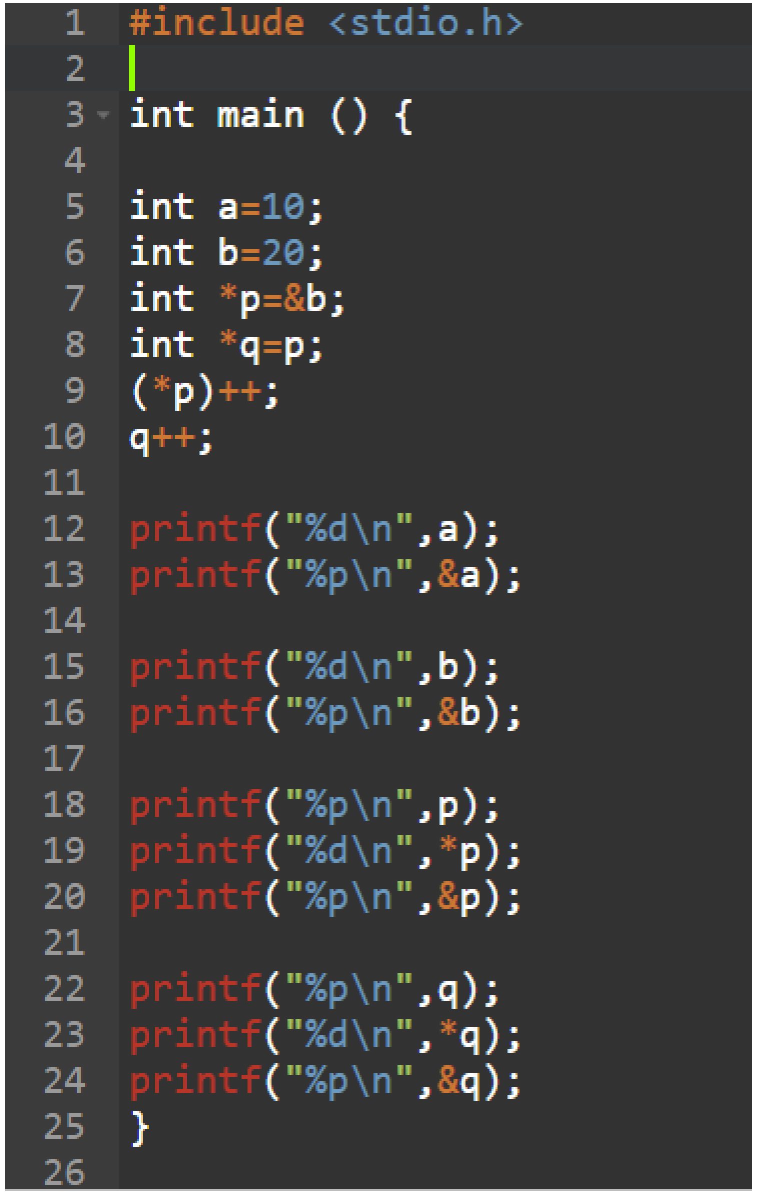
```
15
```

Q01: Use memory sketches provided as a guidance, so as you trace code, you will update values accordingly. **What do you see if you print?**

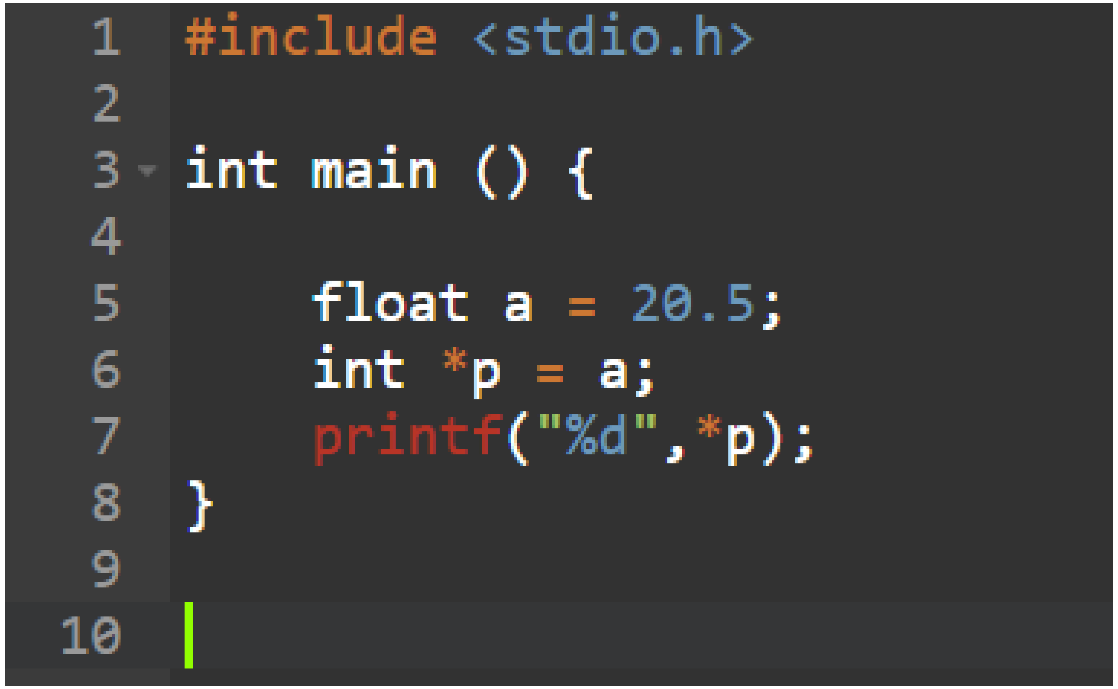
```
int main(void)
{
    int a=10;
    int b=20;
    int *p=&b;
    int *q=p;
    (*p)++;
    q++;
}
```

q	1000	
p	1004	
b	1008	
a	1012	

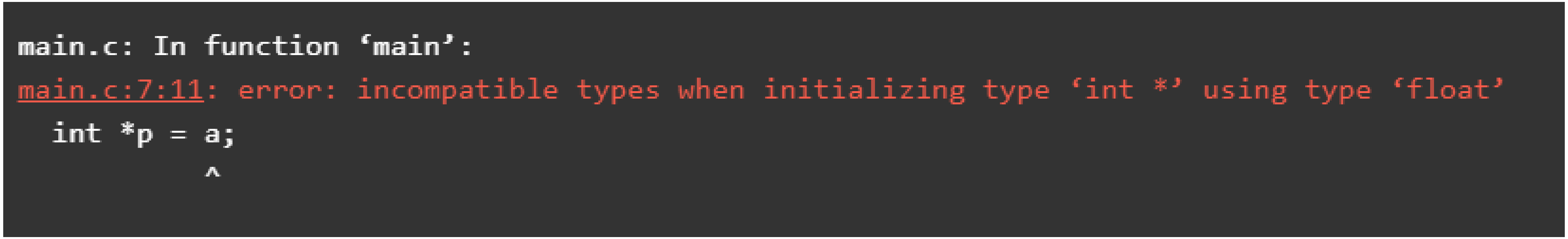
Identifier	a	&a	b	&b	p	*p	&p	q	*q	&q
Value	10	1012	~~20~~ 21	1008	1008	~~20~~ 21	1004	-~~1008~~ 1012	10	1000



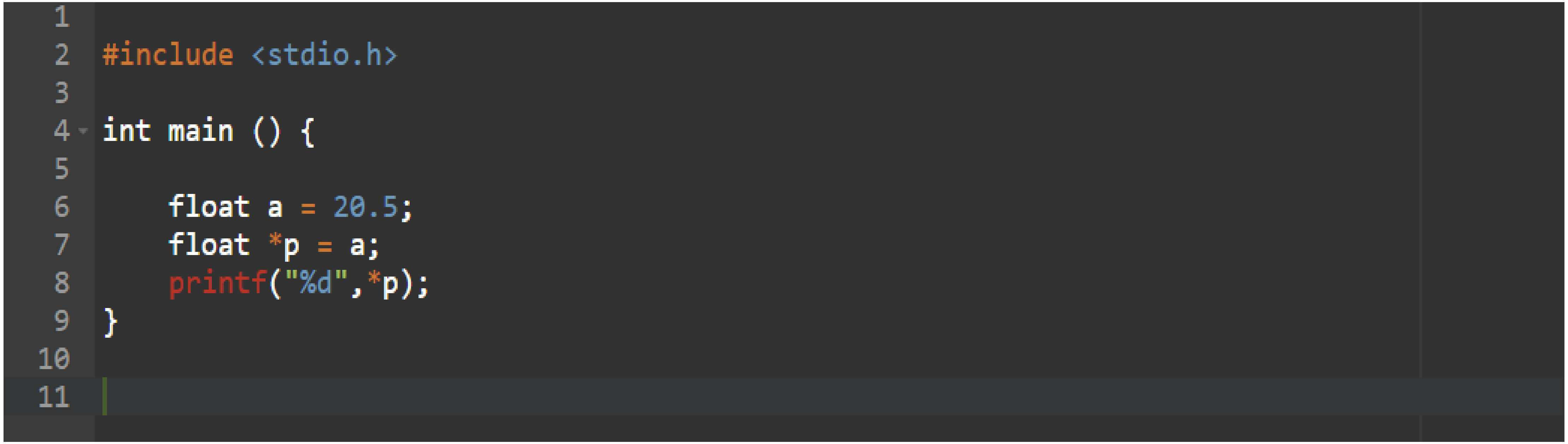
```
#include <stdio.h>

int main () {

int a=10;
int b=20;
int *p=&b;
int *q=p;
(*p)++;
q++;

printf("%d\n",a);
printf("%p\n",&a);

printf("%d\n",b);
printf("%p\n",&b);

printf("%p\n",p);
printf("%d\n",*p);
printf("%p\n",&p);

printf("%p\n",q);
printf("%d\n",*q);
printf("%p\n",&q);
}
```



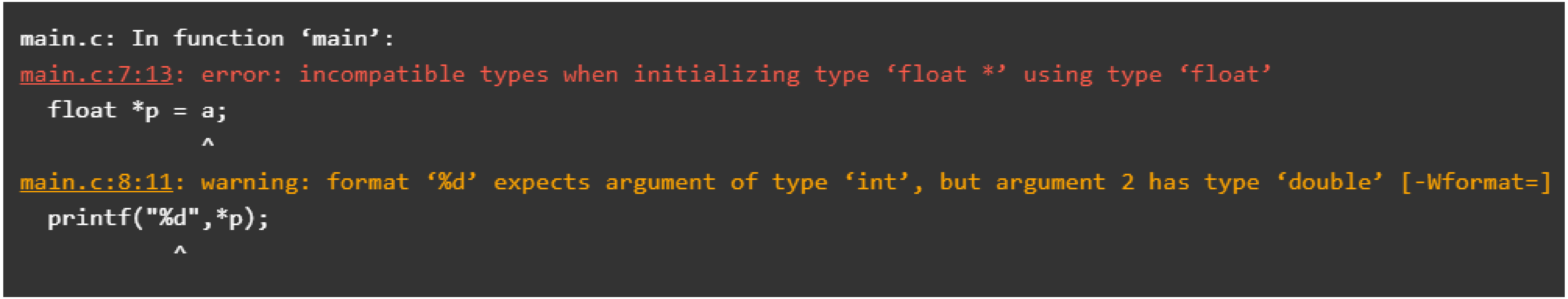
```c
#include <stdio.h>

int main () {

    float a = 20.5;
    int *p = a;
    printf("%d",*p);
}

```



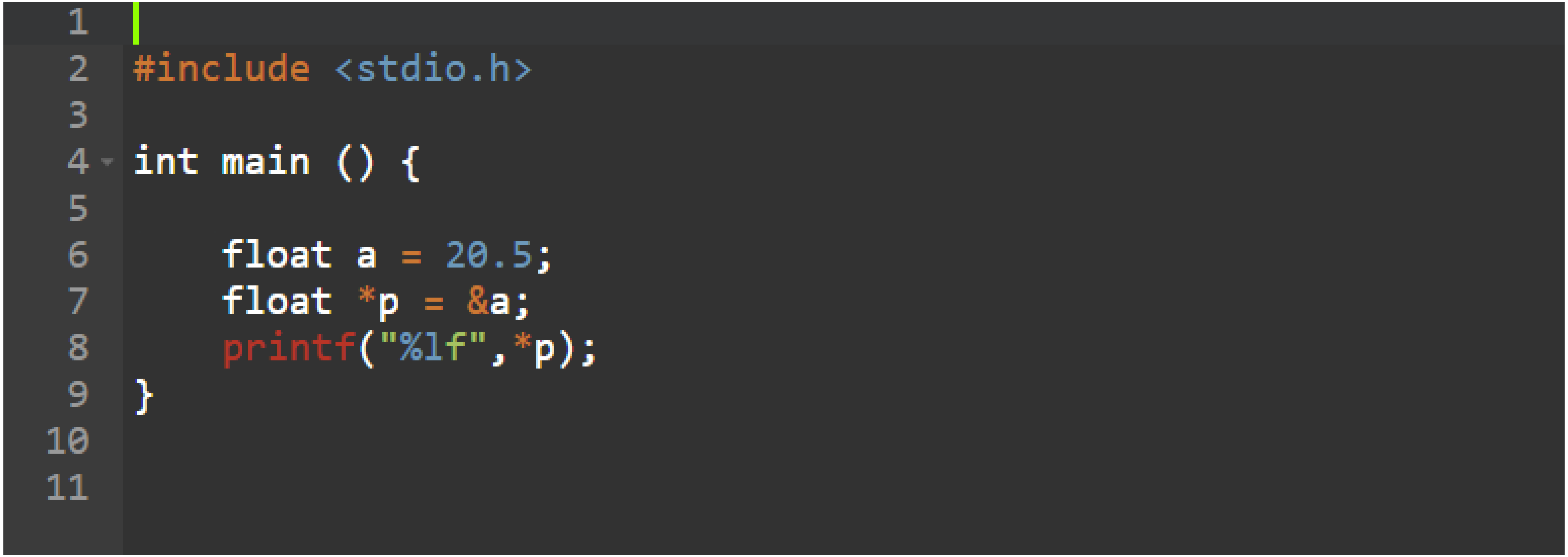
```
main.c: In function ‘main’:
main.c:7:11: error: incompatible types when initializing type ‘int *’ using type ‘float’
  int *p = a;
           ^
```



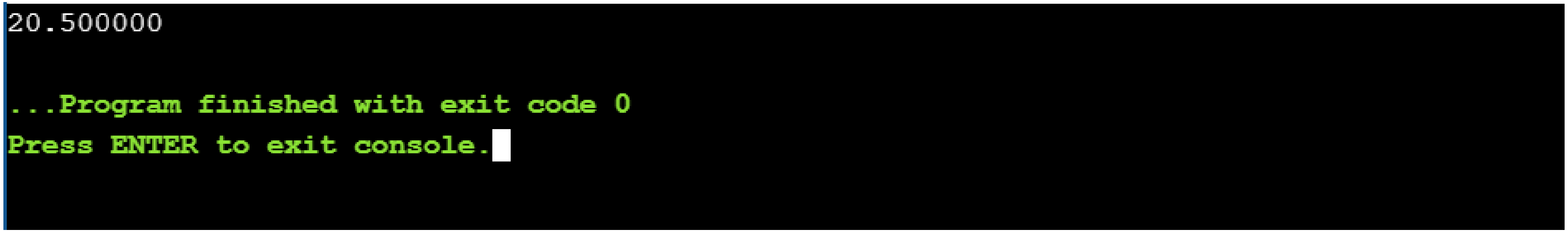
```c

#include <stdio.h>

int main () {

    float a = 20.5;
    float *p = a;
    printf("%d",*p);
}


```



```
main.c: In function ‘main’:
main.c:7:13: error: incompatible types when initializing type ‘float *’ using type ‘float’
  float *p = a;
             ^
main.c:8:11: warning: format ‘%d’ expects argument of type ‘int’, but argument 2 has type ‘double’ [-Wformat=]
  printf("%d",*p);
           ^
```



```c

#include <stdio.h>

int main () {

    float a = 20.5;
    float *p = &a;
    printf("%lf",*p);
}
```

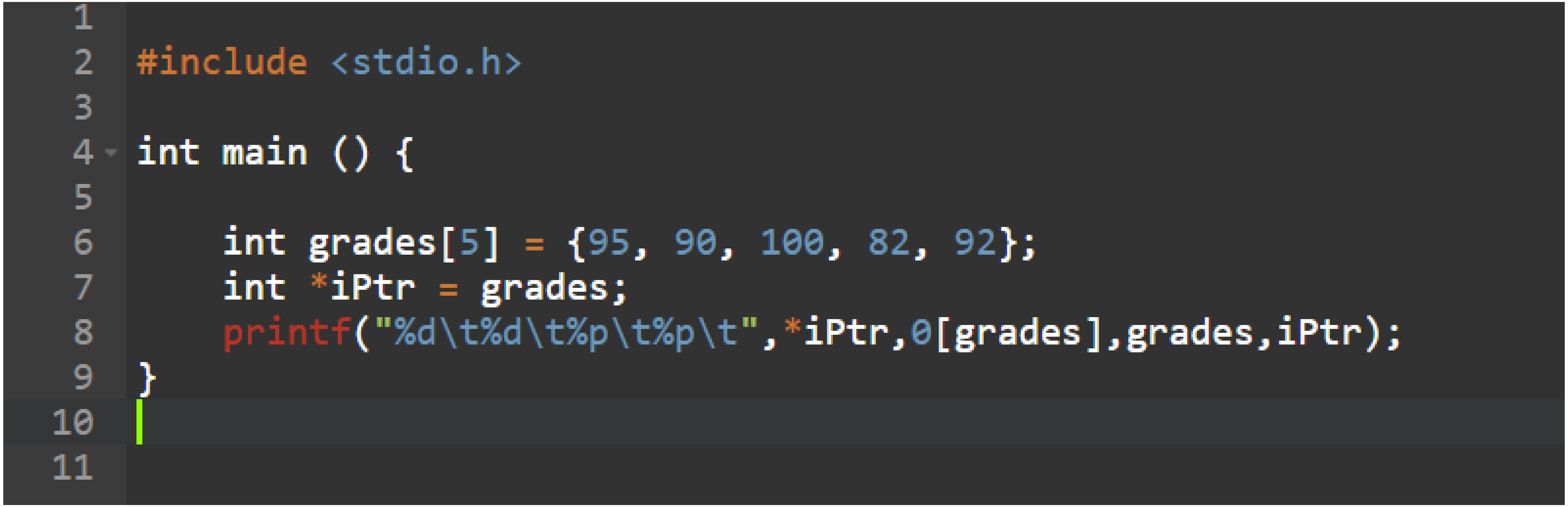


```
20.500000

...Program finished with exit code 0
Press ENTER to exit console.
```

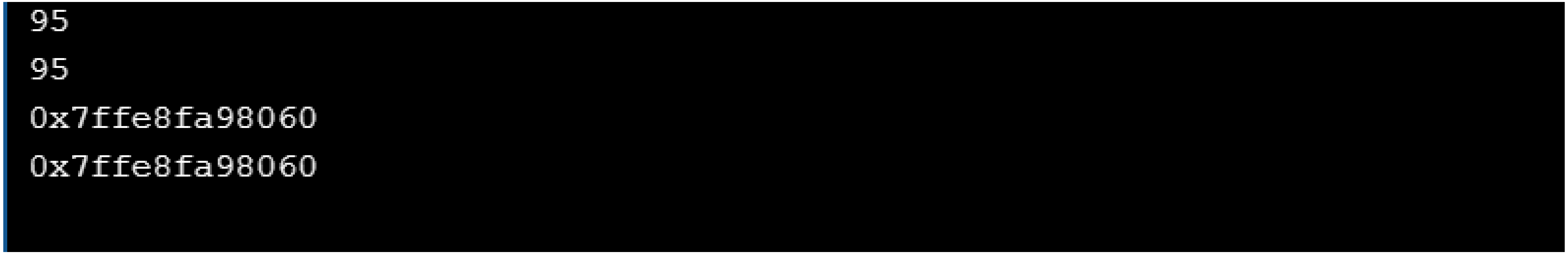
Question: What will be the output of the following code assuming that array begins at location 0X7FEE8FA98060?

```
#include<stdio.h>

int main()
{
    int grades[5] = {95, 90, 100, 82, 92};
    int *iPtr = grades;
    printf(“ %d\n %d\n %p\n %p\n ",*iPtr, 0[grades], grades, iPtr);
}
```

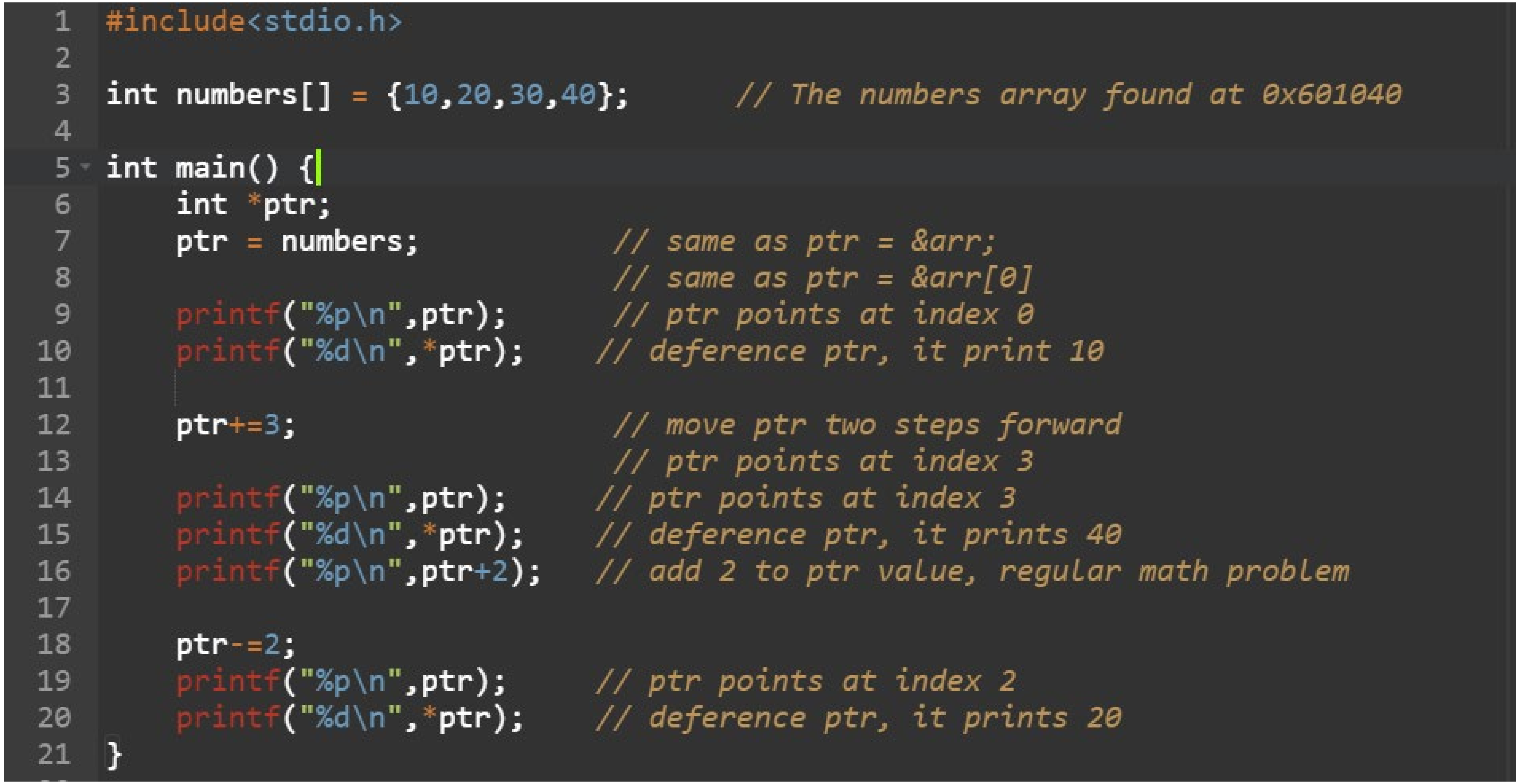
Identifier	*iPtr	0[grades]	grades	iPtr
Value				



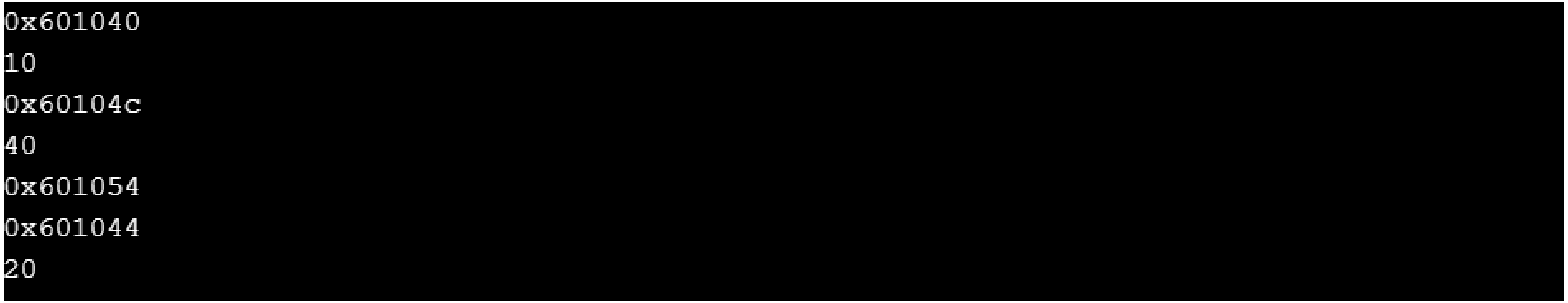
```c

#include <stdio.h>

int main () {

    int grades[5] = {95, 90, 100, 82, 92};
    int *iPtr = grades;
    printf("%d\t%d\t%p\t%p\t",*iPtr,0[grades],grades,iPtr);
}


```



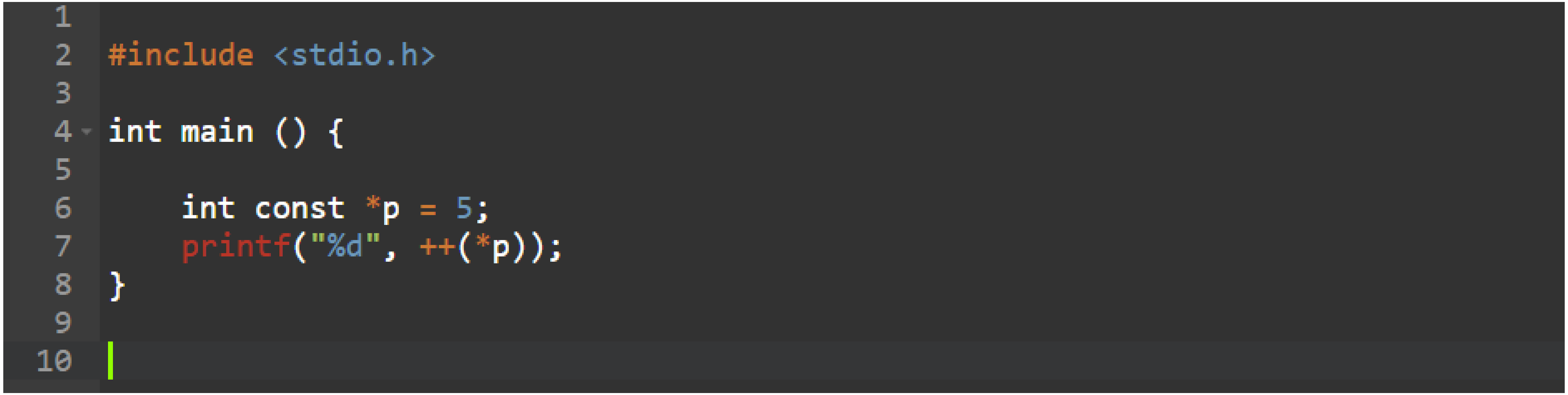
```
95
95
0x7ffe8fa98060
0x7ffe8fa98060
```



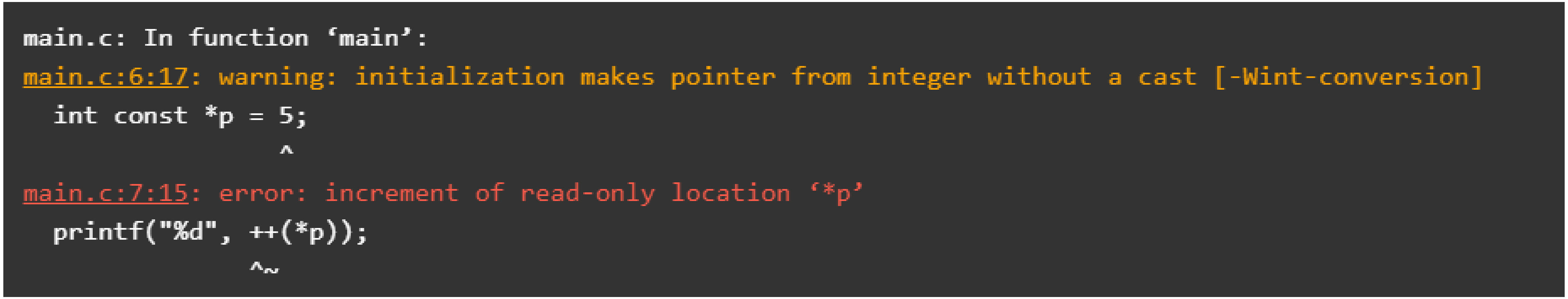
```c
#include<stdio.h>

int numbers[] = {10,20,30,40};      // The numbers array found at 0x601040

int main() {
    int *ptr;
    ptr = numbers;           // same as ptr = &arr;
                             // same as ptr = &arr[0]
    printf("%p\n",ptr);      // ptr points at index 0
    printf("%d\n",*ptr);    // deference ptr, it print 10

    ptr+=3;                  // move ptr two steps forward
                             // ptr points at index 3
    printf("%p\n",ptr);     // ptr points at index 3
    printf("%d\n",*ptr);    // deference ptr, it prints 40
    printf("%p\n",ptr+2);   // add 2 to ptr value, regular math problem

    ptr-=2;
    printf("%p\n",ptr);     // ptr points at index 2
    printf("%d\n",*ptr);    // deference ptr, it prints 20
}
```



```
0x601040
10
0x60104c
40
0x601054
0x601044
20
```

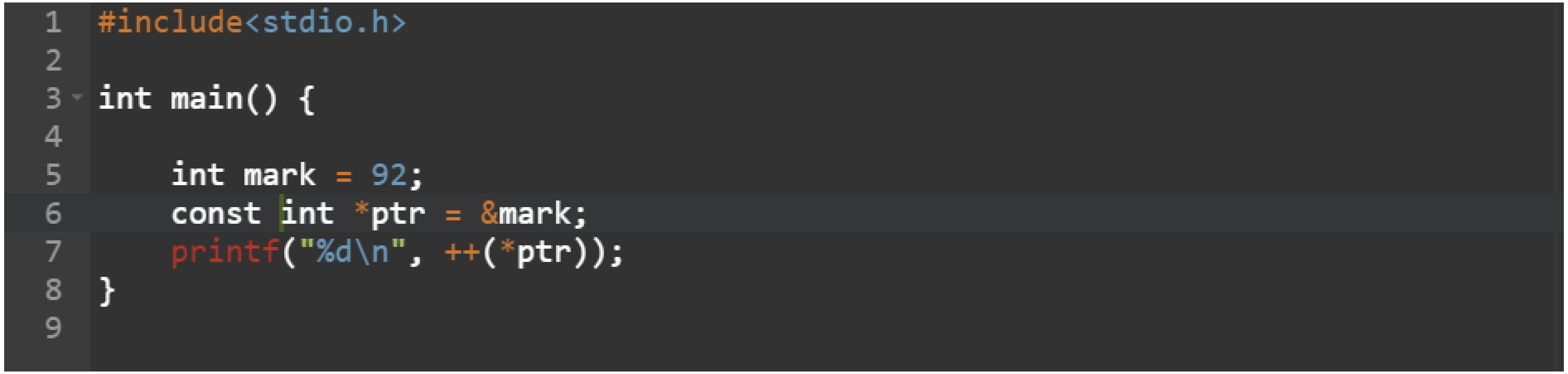


```c

#include <stdio.h>

int main () {

      int const *p = 5;
      printf("%d", ++(*p));
}

```

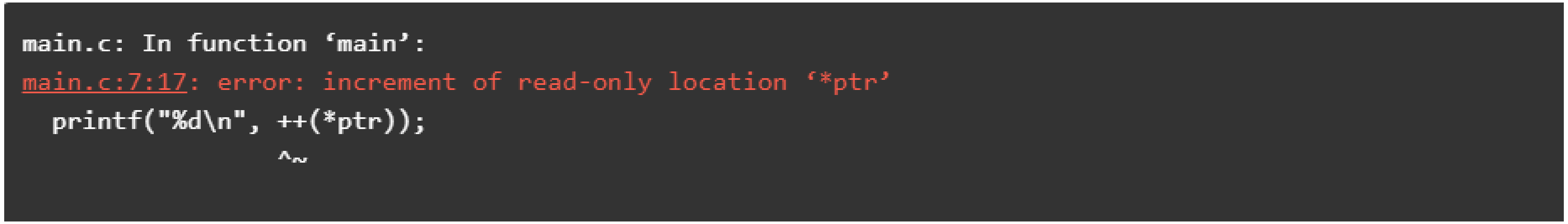


```
main.c: In function ‘main’:
main.c:6:17: warning: initialization makes pointer from integer without a cast [-Wint-conversion]
  int const *p = 5;
                 ^
main.c:7:15: error: increment of read-only location ‘*p’
  printf("%d", ++(*p));
               ^~
```

Note:
ptr is a non-constant pointer to int, and that int is constant

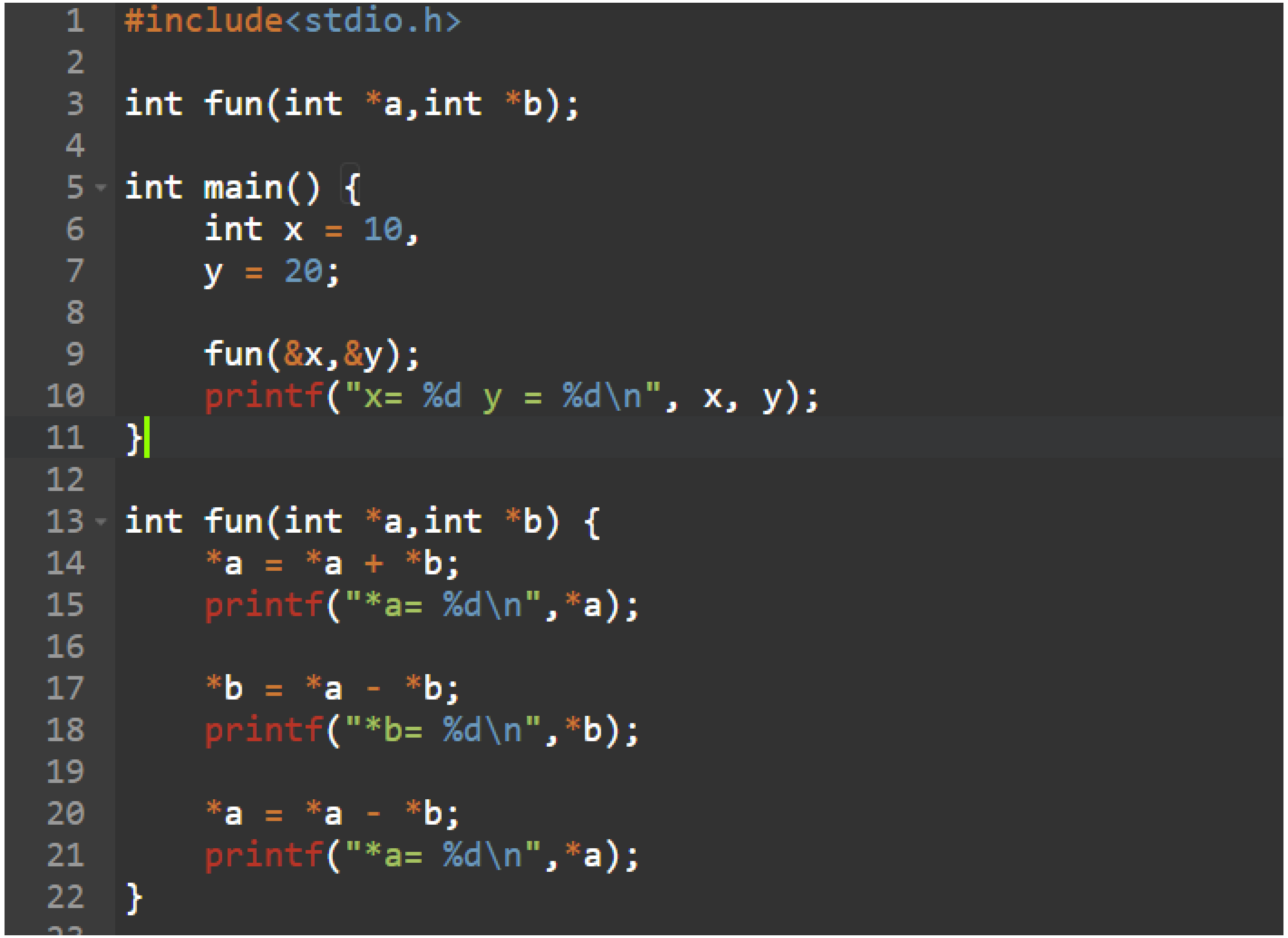


```c
#include<stdio.h>

int main() {

    int mark = 92;
    const int *ptr = &mark;
    printf("%d\n", ++(*ptr));
}

```

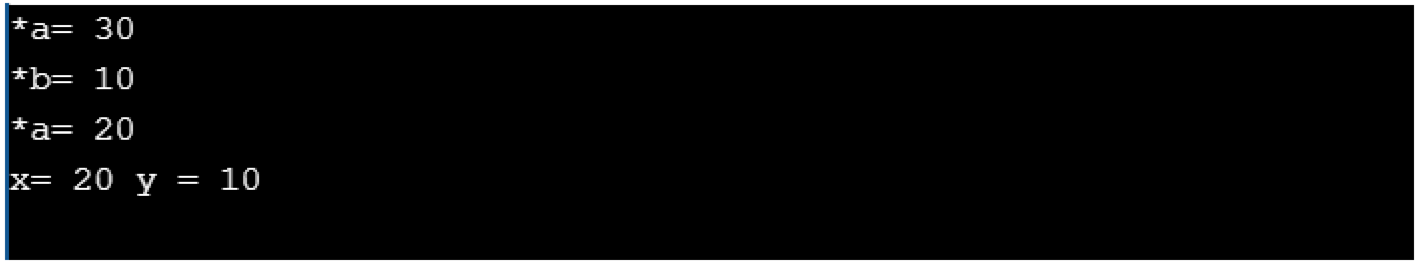


```
main.c: In function ‘main’:
main.c:7:17: error: increment of read-only location ‘*ptr’
  printf("%d\n", ++(*ptr));
                 ^~
```

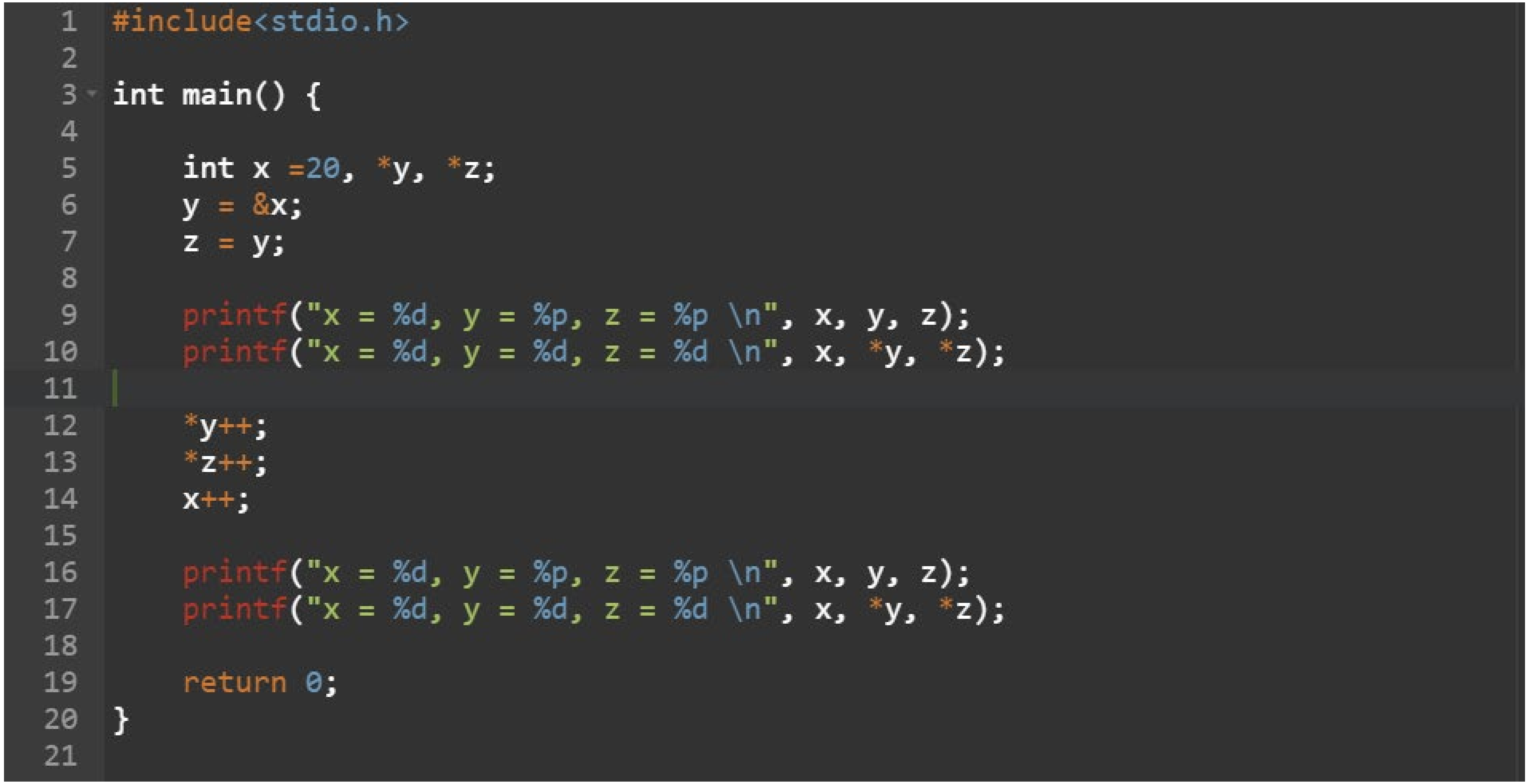
Note: This is another way of declaring constant integer
ptr is a non-constant pointer to int, and that int is constant



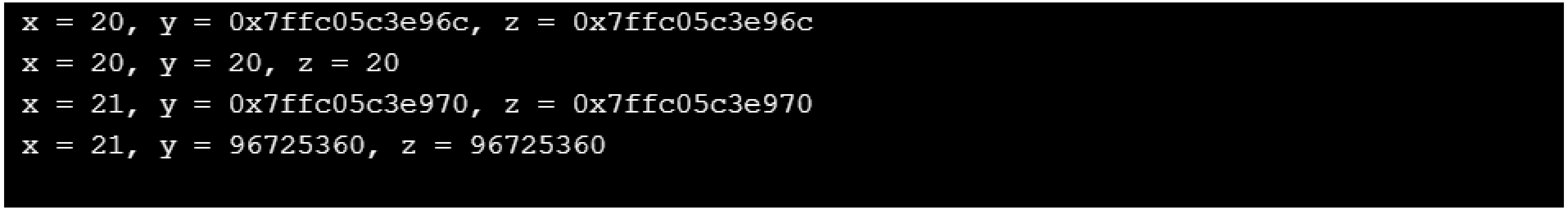
```c
#include<stdio.h>

int fun(int *a,int *b);

int main() {
    int x = 10,
    y = 20;

    fun(&x,&y);
    printf("x= %d y = %d\n", x, y);
}

int fun(int *a,int *b) {
    *a = *a + *b;
    printf("*a= %d\n",*a);

    *b = *a - *b;
    printf("*b= %d\n",*b);

    *a = *a - *b;
    printf("*a= %d\n",*a);
}
```



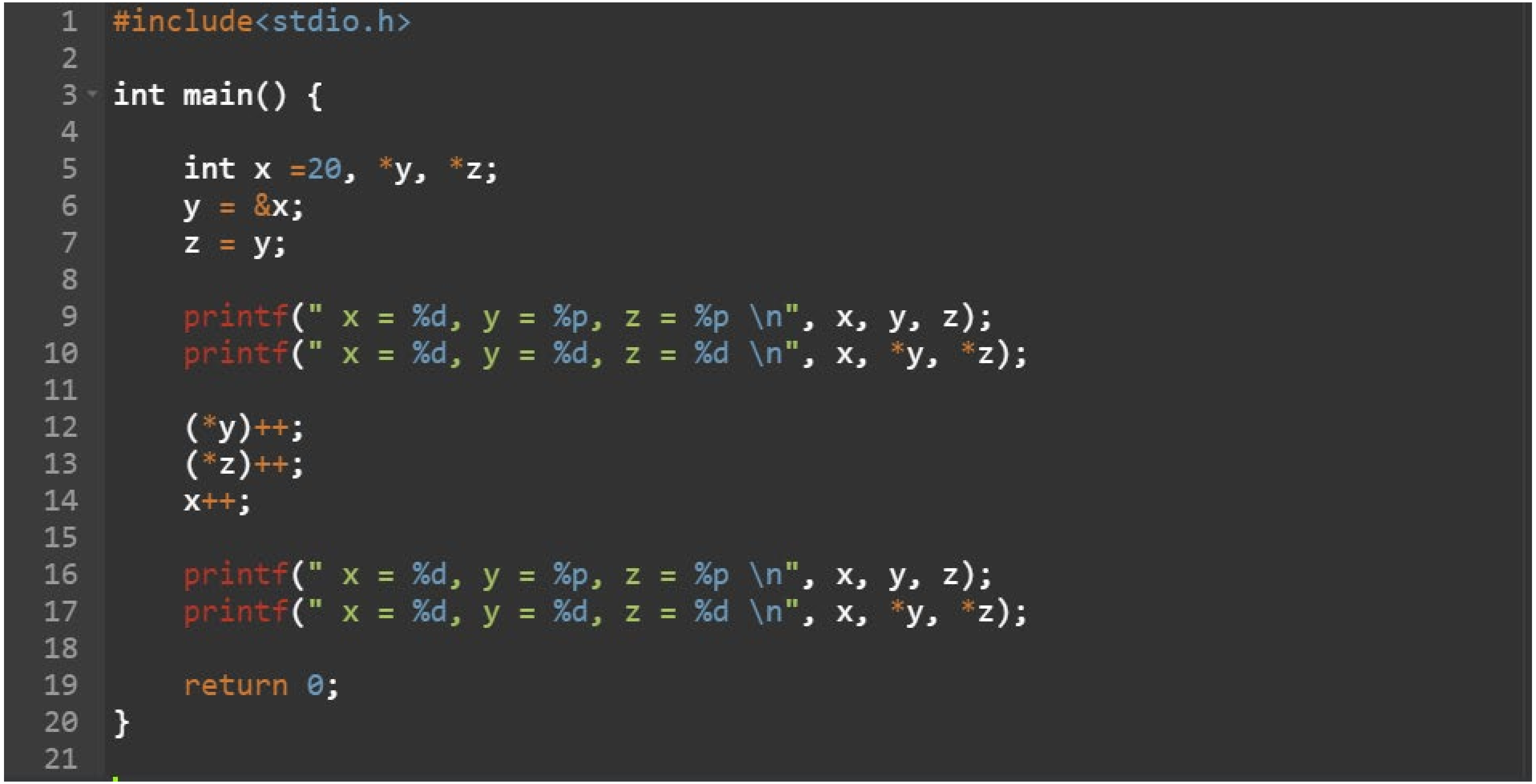
```
*a= 30
*b= 10
*a= 20
x= 20 y = 10
```



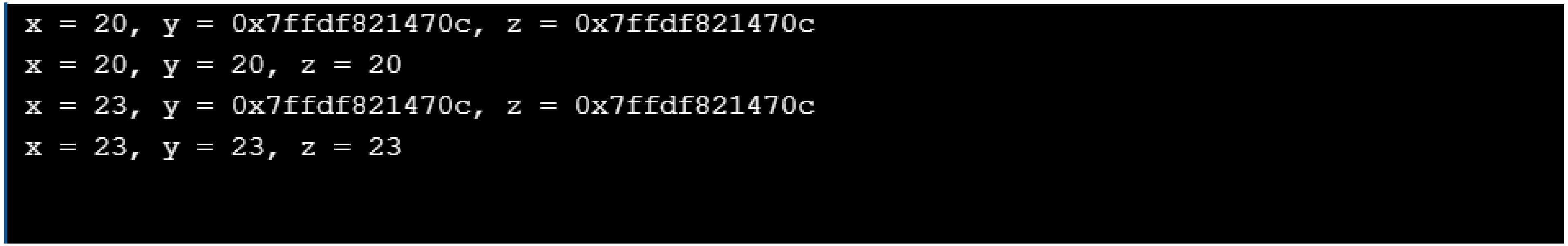
```c
#include<stdio.h>

int main() {

    int x =20, *y, *z;
    y = &x;
    z = y;

    printf("x = %d, y = %p, z = %p \n", x, y, z);
    printf("x = %d, y = %d, z = %d \n", x, *y, *z);

    *y++;
    *z++;
    x++;

    printf("x = %d, y = %p, z = %p \n", x, y, z);
    printf("x = %d, y = %d, z = %d \n", x, *y, *z);

    return 0;
}

```



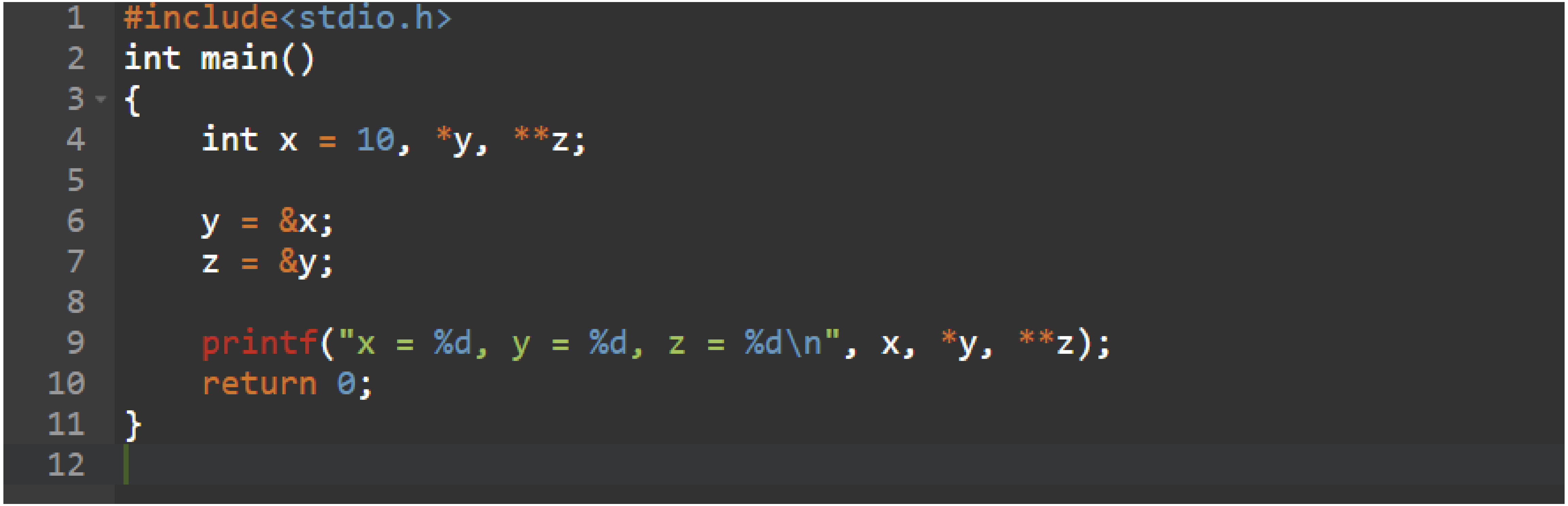
```
x = 20, y = 0x7ffc05c3e96c, z = 0x7ffc05c3e96c
x = 20, y = 20, z = 20
x = 21, y = 0x7ffc05c3e970, z = 0x7ffc05c3e970
x = 21, y = 96725360, z = 96725360
```



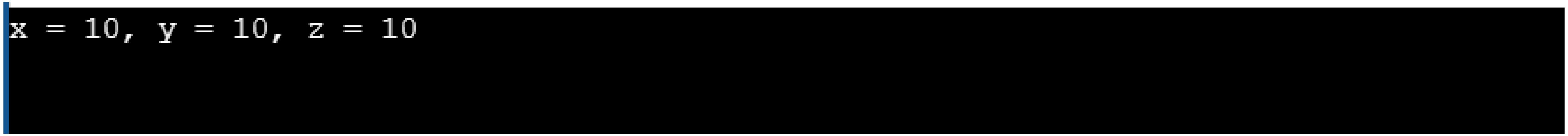
```c
#include<stdio.h>

int main() {

    int x =20, *y, *z;
    y = &x;
    z = y;

    printf(" x = %d, y = %p, z = %p \n", x, y, z);
    printf(" x = %d, y = %d, z = %d \n", x, *y, *z);

    (*y)++;
    (*z)++;
    x++;

    printf(" x = %d, y = %p, z = %p \n", x, y, z);
    printf(" x = %d, y = %d, z = %d \n", x, *y, *z);

    return 0;
}

```



```
 x = 20, y = 0x7ffdf821470c, z = 0x7ffdf821470c
 x = 20, y = 20, z = 20
 x = 23, y = 0x7ffdf821470c, z = 0x7ffdf821470c
 x = 23, y = 23, z = 23
```

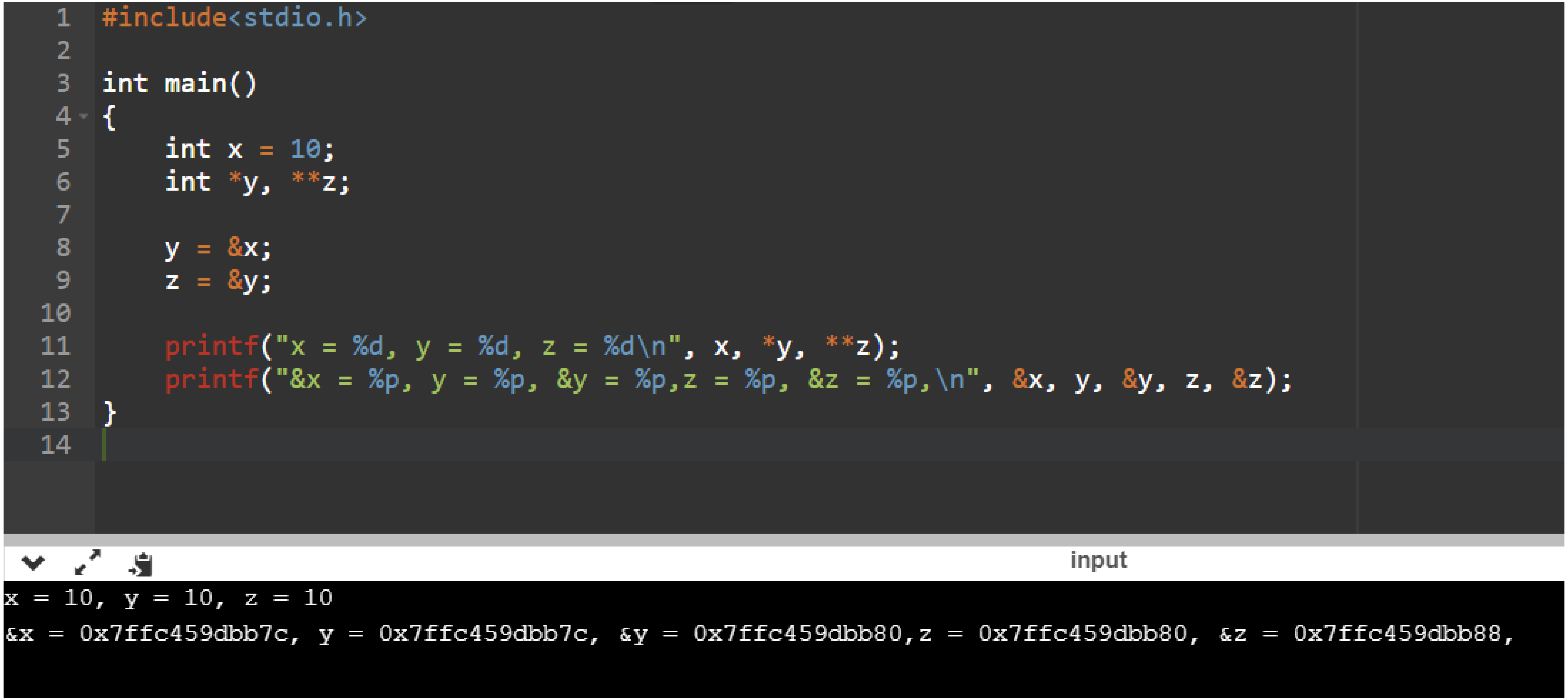


```c
#include<stdio.h>
int main()
{
    int x = 10, *y, **z;

    y = &x;
    z = &y;

    printf("x = %d, y = %d, z = %d\n", x, *y, **z);
    return 0;
}

```



```
x = 10, y = 10, z = 10
```

How many levels of pointers can we have? Here



```c
#include<stdio.h>

int main()
{
    int x = 10;
    int *y, **z;

    y = &x;
    z = &y;

    printf("x = %d, y = %d, z = %d\n", x, *y, **z);
    printf("&x = %p, y = %p, &y = %p,z = %p, &z = %p,\n", &x, y, &y, z, &z);
}

```

```
x = 10, y = 10, z = 10
&x = 0x7ffc459dbb7c, y = 0x7ffc459dbb7c, &y = 0x7ffc459dbb80,z = 0x7ffc459dbb80, &z = 0x7ffc459dbb88,
```

Midterm Cheat Sheet

- This table is all what you need to know/use/refer to during midterm.
- All questions and expected answers are based on 32 bits systems.

Type	Size
char	1
short	2
int	4
long	8
float	4
double	8
void*	4

Note: Section 08 (C Strings and Characters) has been excluded from midterm.