CSCI 476: Computer Security

Hashing (Part 3)

Reese Pearsall

Spring 2023

Announcements

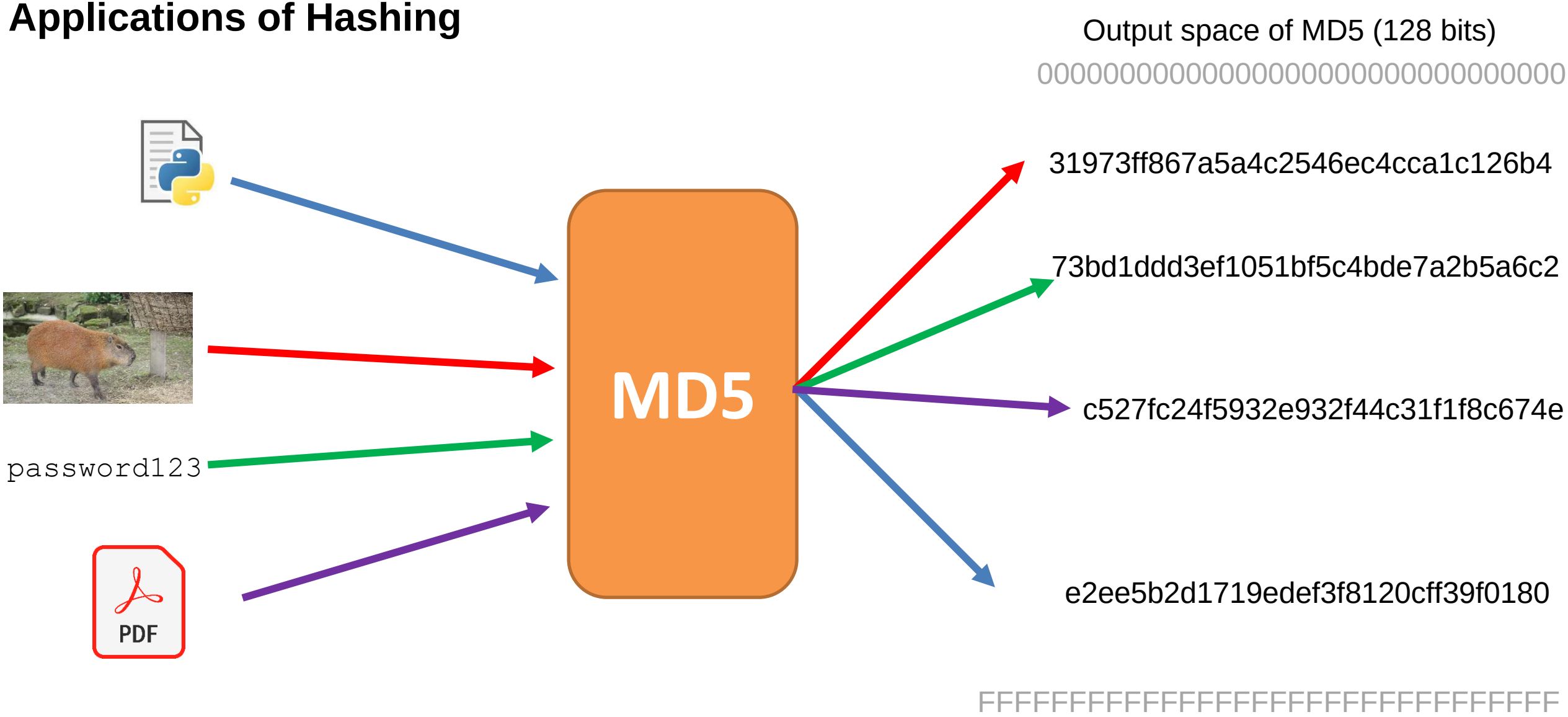
Lab 8 due Wednesday April 19th

Research Project due April 23rd

Lab 9 due Sunday April 30th

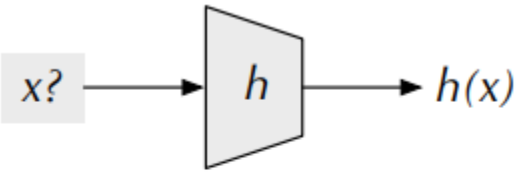
Final Lab will be posted sometime after the research project
→Due during finals week
→You can earn extra credit for the final lab by attending lecture on 4/24, 4/26, 4/28, 5/1

Applications of Hashing

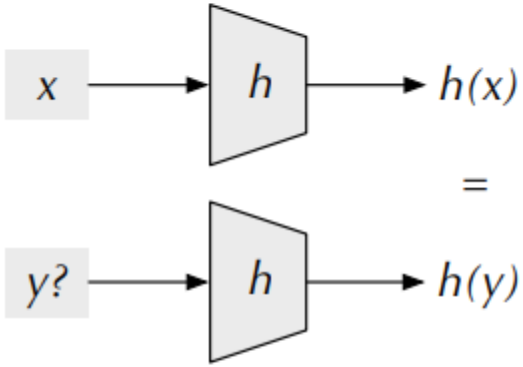


Hash Functions Properties

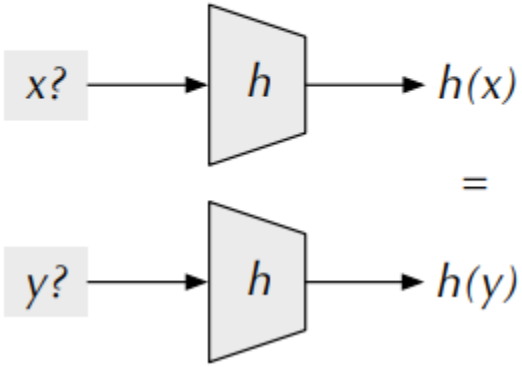
- **Preimage Resistance ("One-Way")**
  Given h(x) = z, hard to find x
  (or any input that hashes to z for that matter)



- **Second Preimage Resistance**
  Given x and h(x), hard to find y s.t. h(x) = h(y)



- Collision Resistance (or, ideally, "Collision Free")
  Difficult to find x and y s.t. hash(x) = hash(y)



Hash Collisions

Goal: Create two **different files** with the **same md5 hash**

Our **ultimate goal** would be to create two executables (one benign, one malicious) with the same hash

(This is difficult to do, but we will show that it can theoretically happen)

Motivation

Forging public-key certificates

md5 () == md5 ()

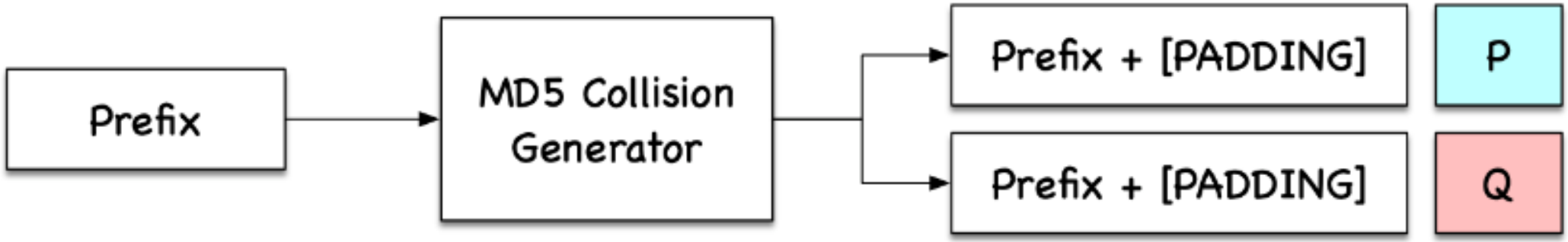
- Assume two certificate requests for www.example.com and www.attacker.com have same hash due to a collision
- CA signing of either request would be equivalent
- Attacker can get certificate signed for www.example.com without owning it!

Integrity of Programs

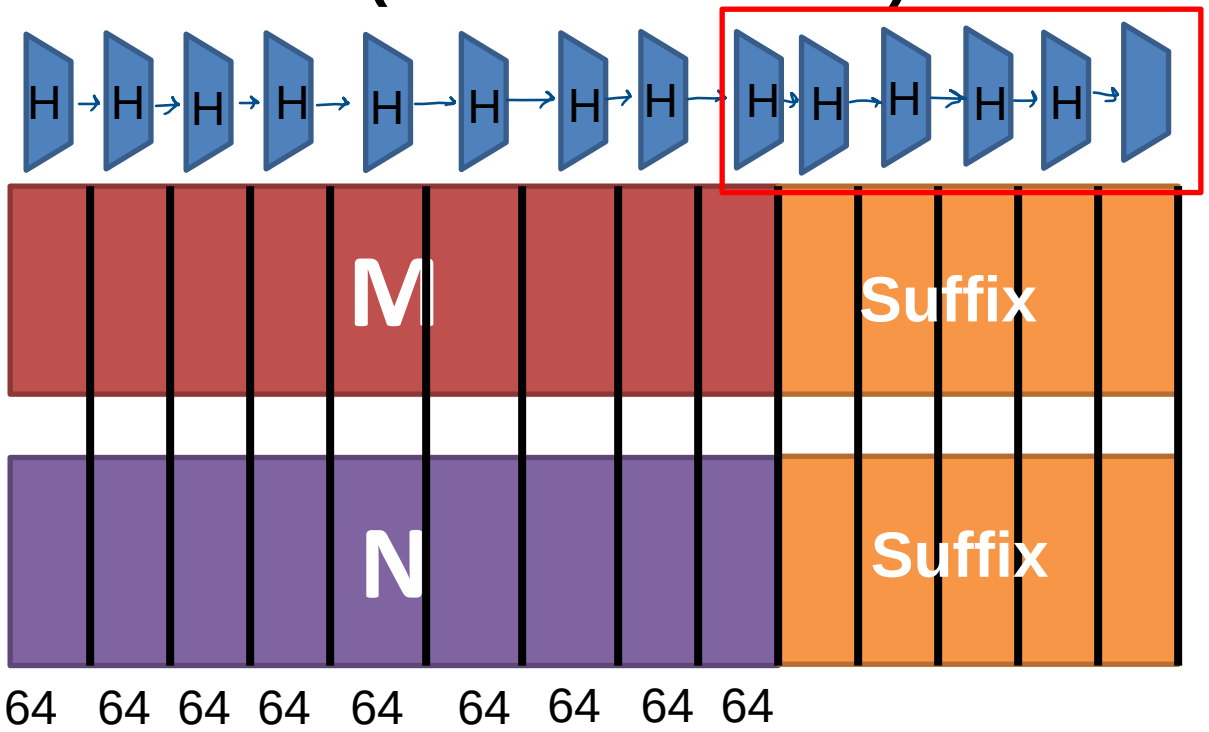
md5 () == md5 ()

- Ask CA to sign a legitimate program's hash
- Attacker creates a malicious program with same hash
- The certificate for legitimate program is also valid for malicious version

Hash Collisions (MD5collgen)



Hash Collisions (Suffix Extension)



If we append the same suffix, then this computation will also be the exact same for M and N

```
[11/17/22]seed@VM:~/.../07_hash$ echo "suffix" > suffix.txt
[11/17/22]seed@VM:~/.../07_hash$ cat out1.bin suffix.txt > out1suffix.bin
][11/17/22]seed@VM:~/.../07_hash$ cat out2.bin suffix.txt > out2suffix.bin
```

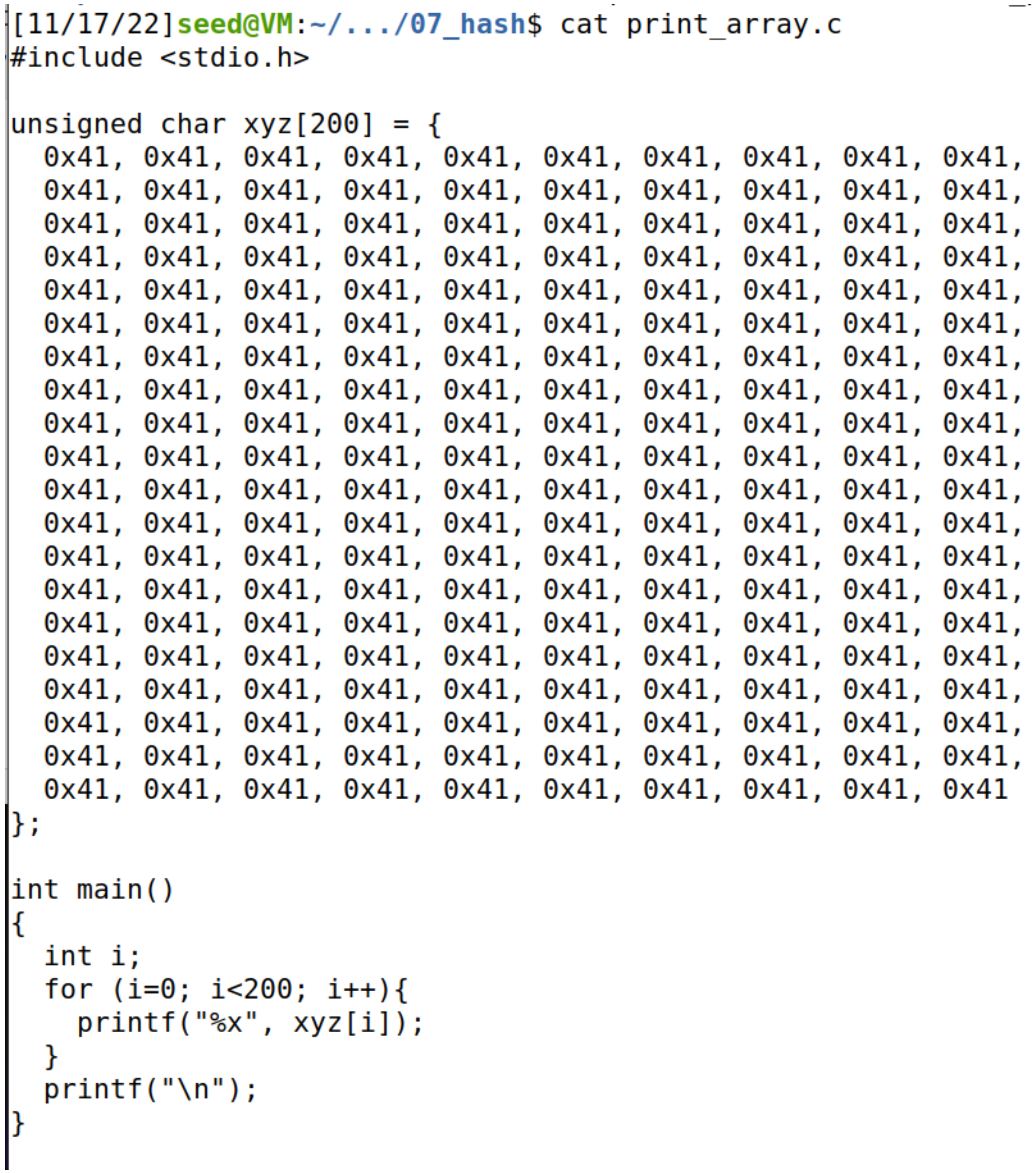
$$H(m) == H(n)$$

$$H(m \,||\, s) == H(n \,||\, s)$$

s = shared suffix

```
[11/17/22]seed@VM:~/.../07_hash$ md5sum out1suffix.bin
a63075af11518048cff11bf3d11a5462  out1suffix.bin
[11/17/22]seed@VM:~/.../07_hash$ md5sum out2suffix.bin
a63075af11518048cff11bf3d11a5462 _out2suffix.bin
```

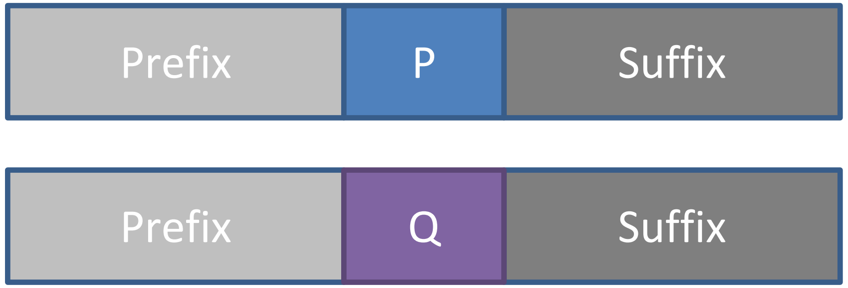
Hash Collisions (Generating Two executable files with the same MD5 hash)



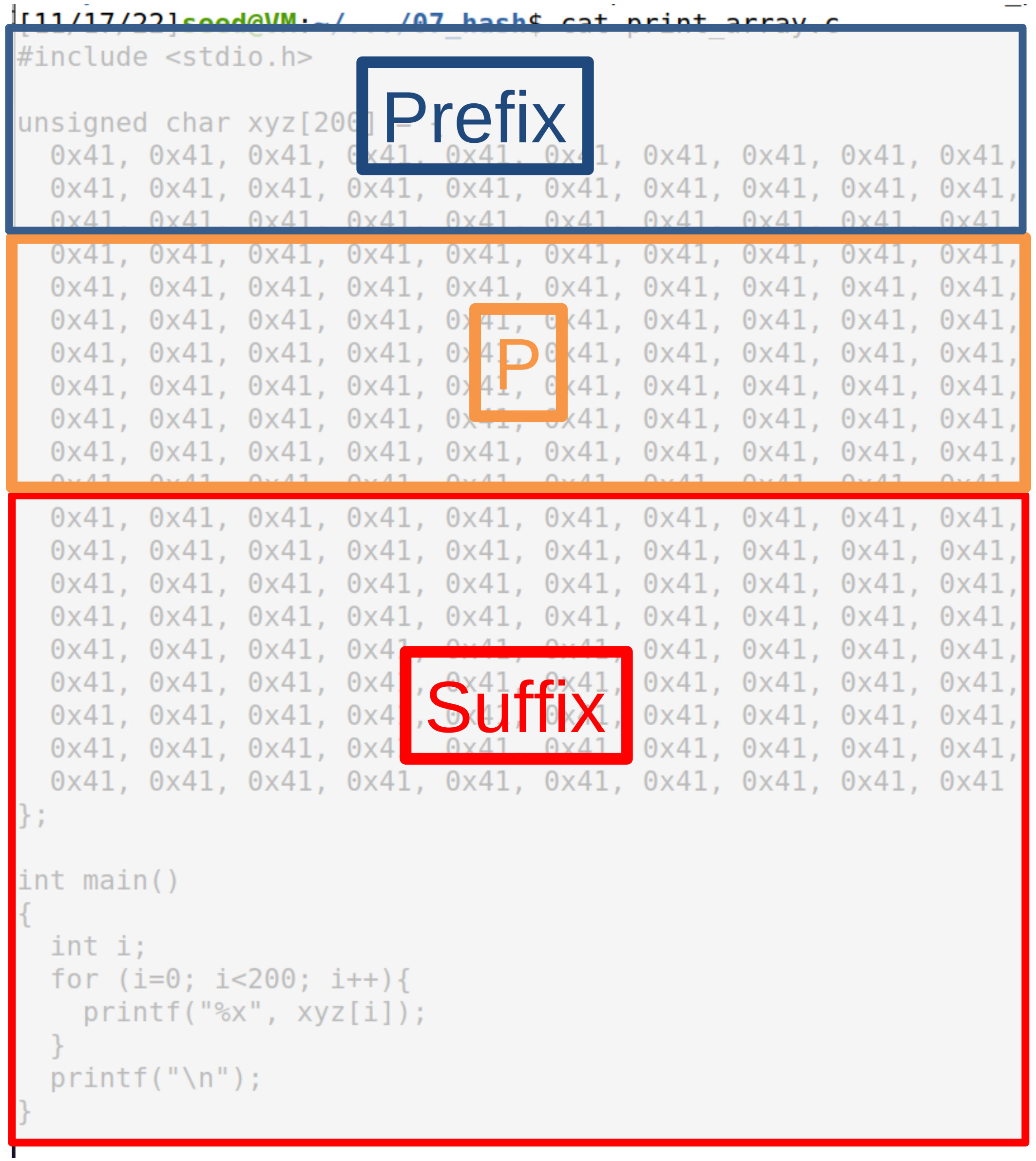
```
[11/17/22]seed@VM:~/.../07_hash$ cat print_array.c
#include <stdio.h>

unsigned char xyz[200] = {
  0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41,
  0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41,
  0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41,
  0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41,
  0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41,
  0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41,
  0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41,
  0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41,
  0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41,
  0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41,
  0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41,
  0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41,
  0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41,
  0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41,
  0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41,
  0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41,
  0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41,
  0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41,
  0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41,
  0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41
};

int main()
{
  int i;
  for (i=0; i<200; i++){
    printf("%x", xyz[i]);
  }
  printf("\n");
}
```

This is a program that will print out the contents of an array

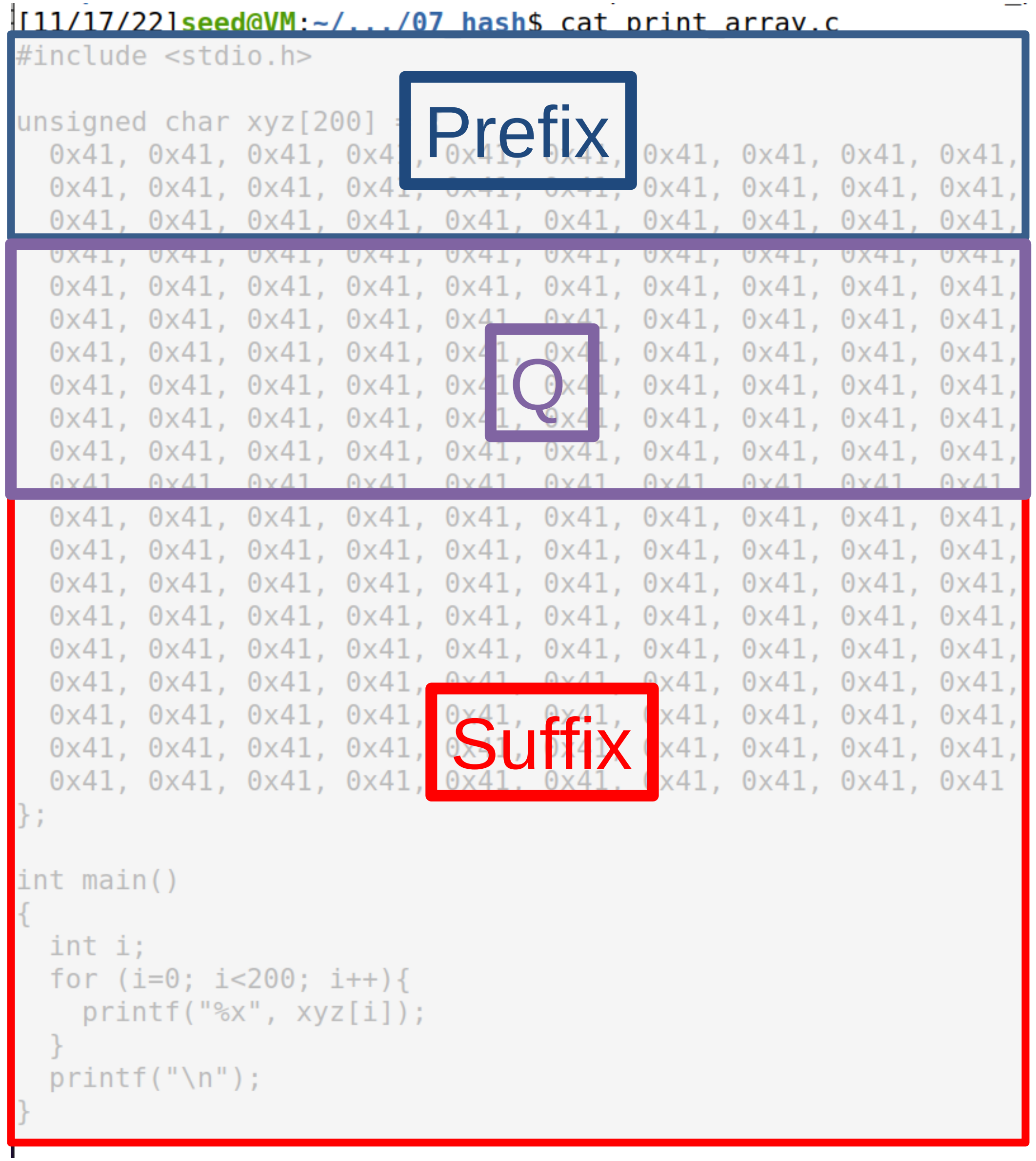
We will create two variants of this program, but the program will have the same hash



Hash Collisions (Generating Two executable files with the same MD5 hash)



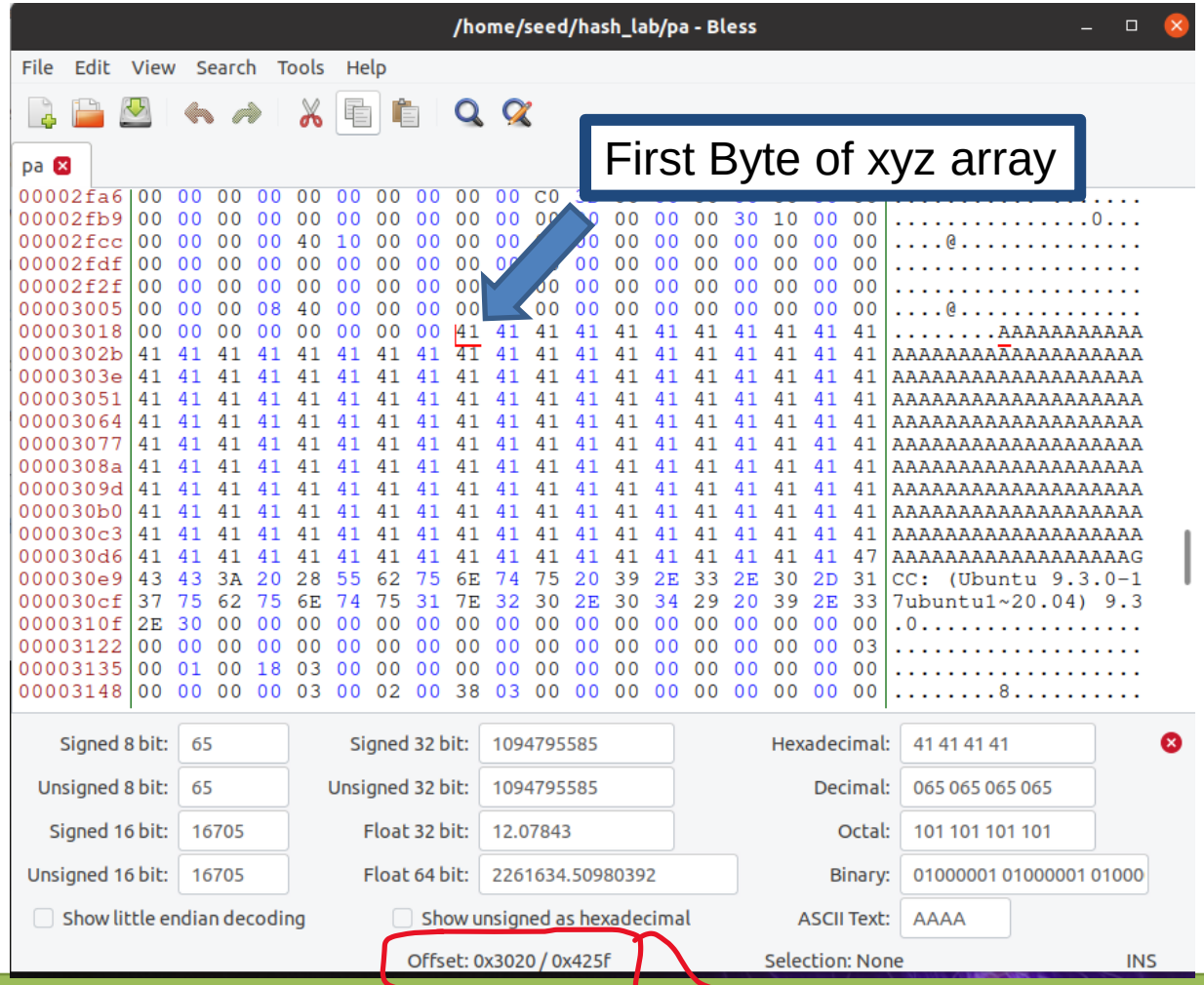
```
[11/17/22]seed@VM:~/.../07_hash$ cat print_array.c
#include <stdio.h>

unsigned char xyz[200] = {
  0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41,
  0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41,
  0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41,
  0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41,
  0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41,
  0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41,
  0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41,
  0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41,
  0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41,
  0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41,
  0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41,
  0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41,
  0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41,
  0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41,
  0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41,
  0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41,
  0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41,
  0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41,
  0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41,
  0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41
};

int main()
{
  int i;
  for (i=0; i<200; i++){
    printf("%x", xyz[i]);
  }
  printf("\n");
}
```



```
[11/17/22]seed@VM:~/.../07_hash$ cat print_array.c
#include <stdio.h>

unsigned char xyz[200] = {
  0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41,
  0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41,
  0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41,
  0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41,
  0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41,
  0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41,
  0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41,
  0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41,
  0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41,
  0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41,
  0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41,
  0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41,
  0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41,
  0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41,
  0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41,
  0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41,
  0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41,
  0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41,
  0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41,
  0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41
};

int main()
{
  int i;
  for (i=0; i<200; i++){
    printf("%x", xyz[i]);
  }
  printf("\n");
}
```

Hash Collisions (Generating Two executable files with the same MD5 hash but behave very differently)

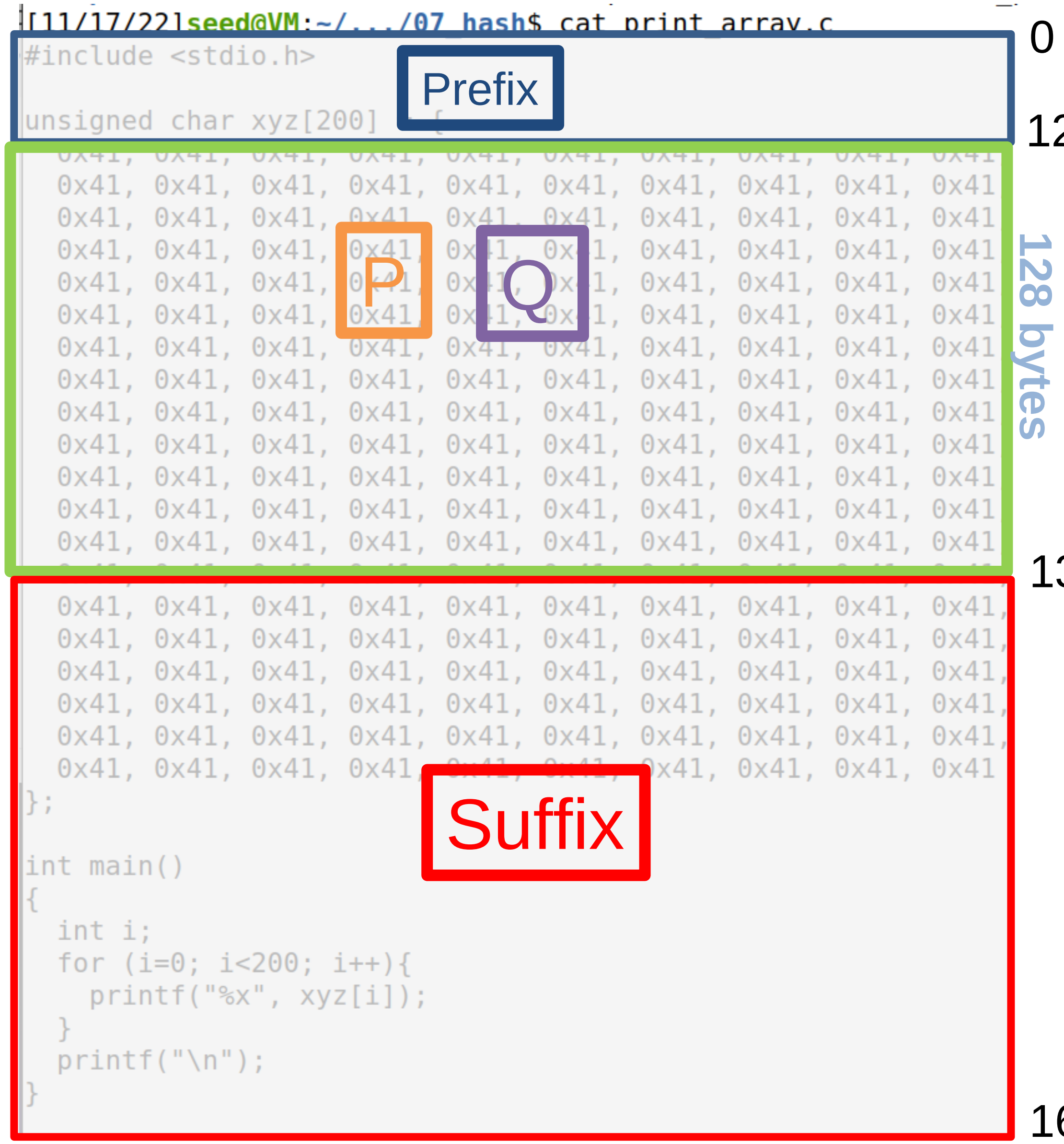
```
[11/17/22]seed@VM:~/hash_lab$ gcc print_array.c -o pa
[11/17/22]seed@VM:~/hash_lab$ bless pa
```



We can find where xyz begins in our program easily, because we filled it with A's

Start of XYZ = 0x3020 (Hexadecimal)
12320 (decimal)

Task 4 on the lab



```
[11/17/22]seed@VM:~/.../07_hash$ cat print_array.c
#include <stdio.h>

unsigned char xyz[200] = {
  0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41
  0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41
  0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41
  0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41
  0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41
  0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41
  0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41
  0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41
  0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41
  0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41
  0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41
  0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41
  0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41
  0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41,
  0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41,
  0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41,
  0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41,
  0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41,
  0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41
};

int main()
{
  int i;
  for (i=0; i<200; i++){
    printf("%x", xyz[i]);
  }
  printf("\n");
}
```

0

12320

13448

16992 (size of executable)

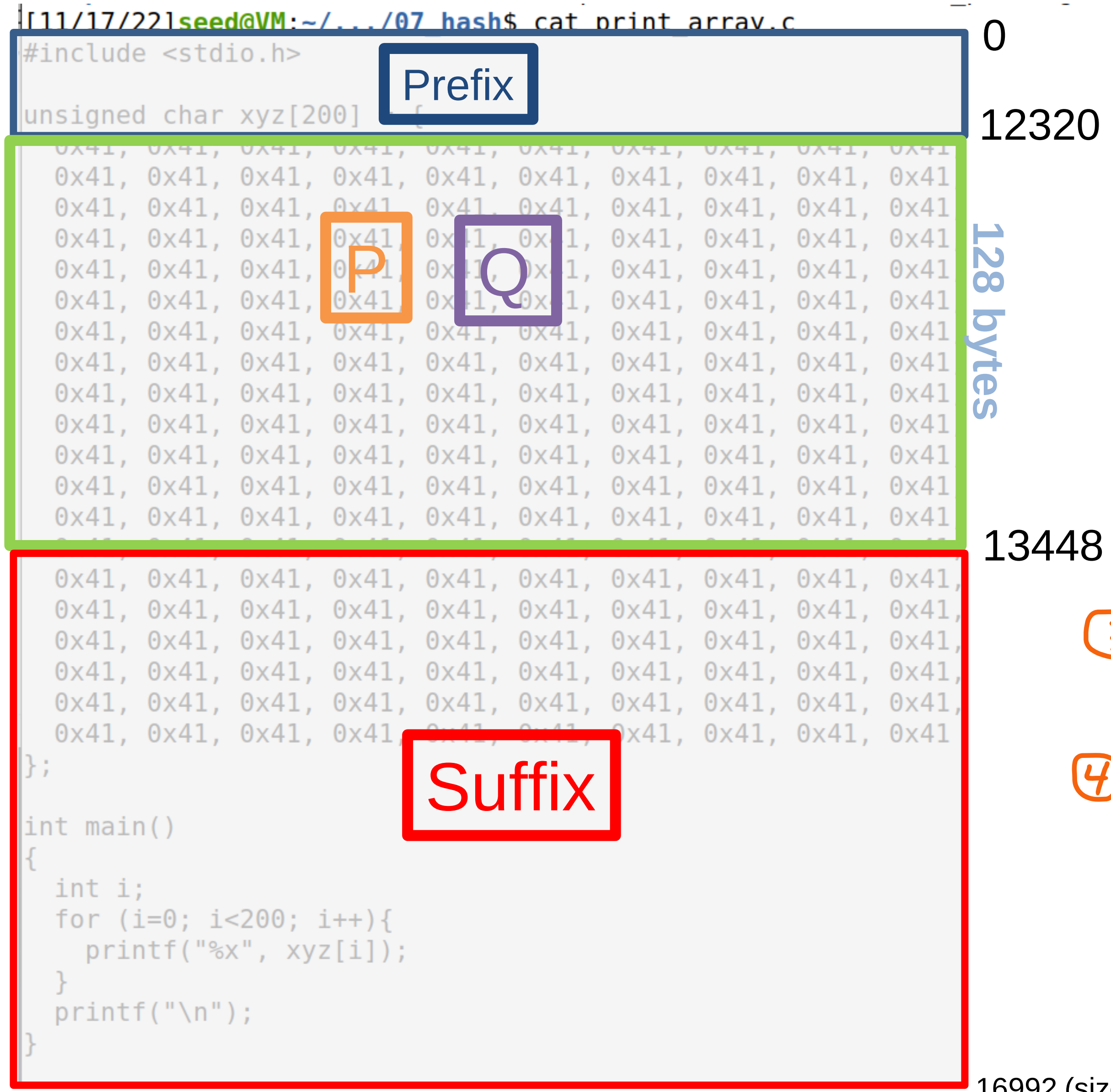
Our prefix will be bytes 0-12320 of the program!

We want our **P** and **Q** to be 128 bytes

Why 128?
→ Multiple of 64
→ Wont overflow an array of size 200

Therefore, our suffix will begin at byte # 12320 + 128 = **13448**

Task 4 on the lab

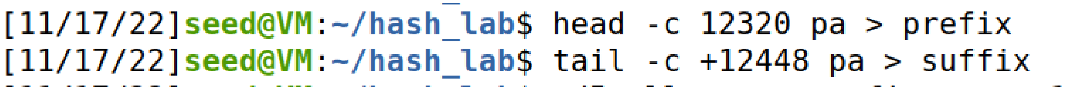


```
[11/17/22]seed@VM:~/.../07_hash$ cat print_array.c
#include <stdio.h>

unsigned char xyz[200] = {
  0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41
  0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41
  0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41
  0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41
  0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41
  0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41
  0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41
  0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41
  0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41
  0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41
  0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41
  0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41
  0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41
  0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41,
  0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41,
  0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41,
  0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41,
  0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41,
  0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41
};

int main()
{
  int i;
  for (i=0; i<200; i++){
    printf("%x", xyz[i]);
  }
  printf("\n");
}
```

16992 (size of executable)

①

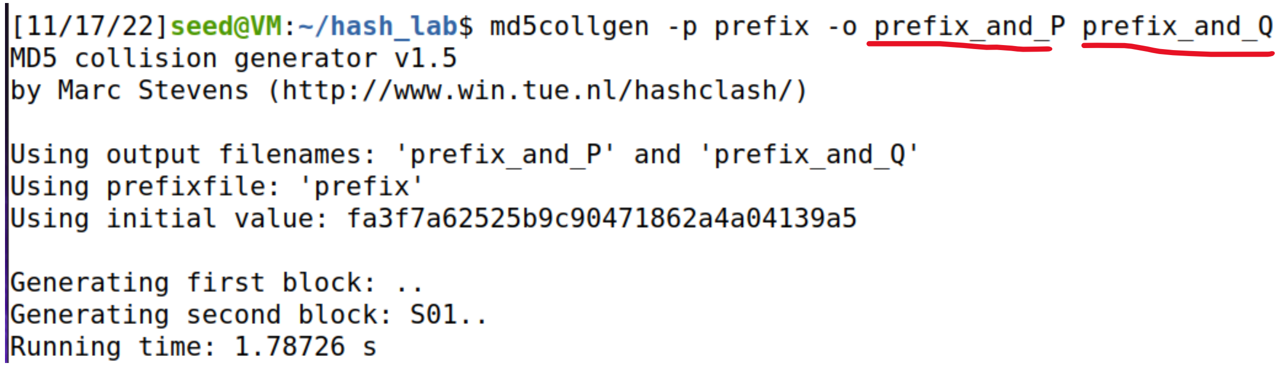
Get contents of prefix and suffix



```
[11/17/22]seed@VM:~/hash_lab$ head -c 12320 pa > prefix
[11/17/22]seed@VM:~/hash_lab$ tail -c +12448 pa > suffix
```

②

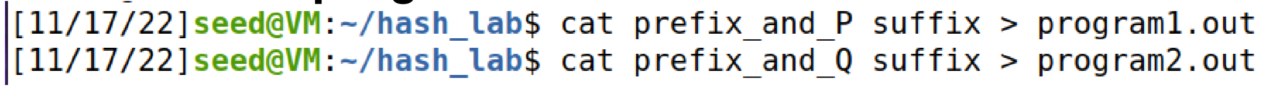
Use collision tool to get (prefix + P) and (prefix + Q)



```
[11/17/22]seed@VM:~/hash_lab$ md5collgen -p prefix -o prefix_and_P prefix_and_Q
MD5 collision generator v1.5
by Marc Stevens (http://www.win.tue.nl/hashclash/)

Using output filenames: 'prefix_and_P' and 'prefix_and_Q'
Using prefixfile: 'prefix'
Using initial value: fa3f7a62525b9c90471862a4a04139a5

Generating first block: ..
Generating second block: S01..
Running time: 1.78726 s
```

③

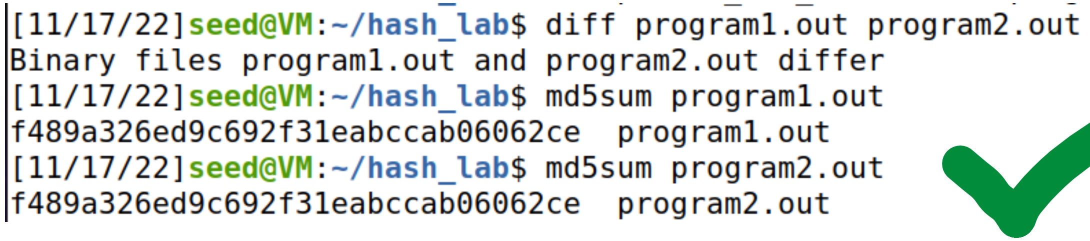
Add suffix to programs



```
[11/17/22]seed@VM:~/hash_lab$ cat prefix_and_P suffix > program1.out
[11/17/22]seed@VM:~/hash_lab$ cat prefix_and_Q suffix > program2.out
```

④

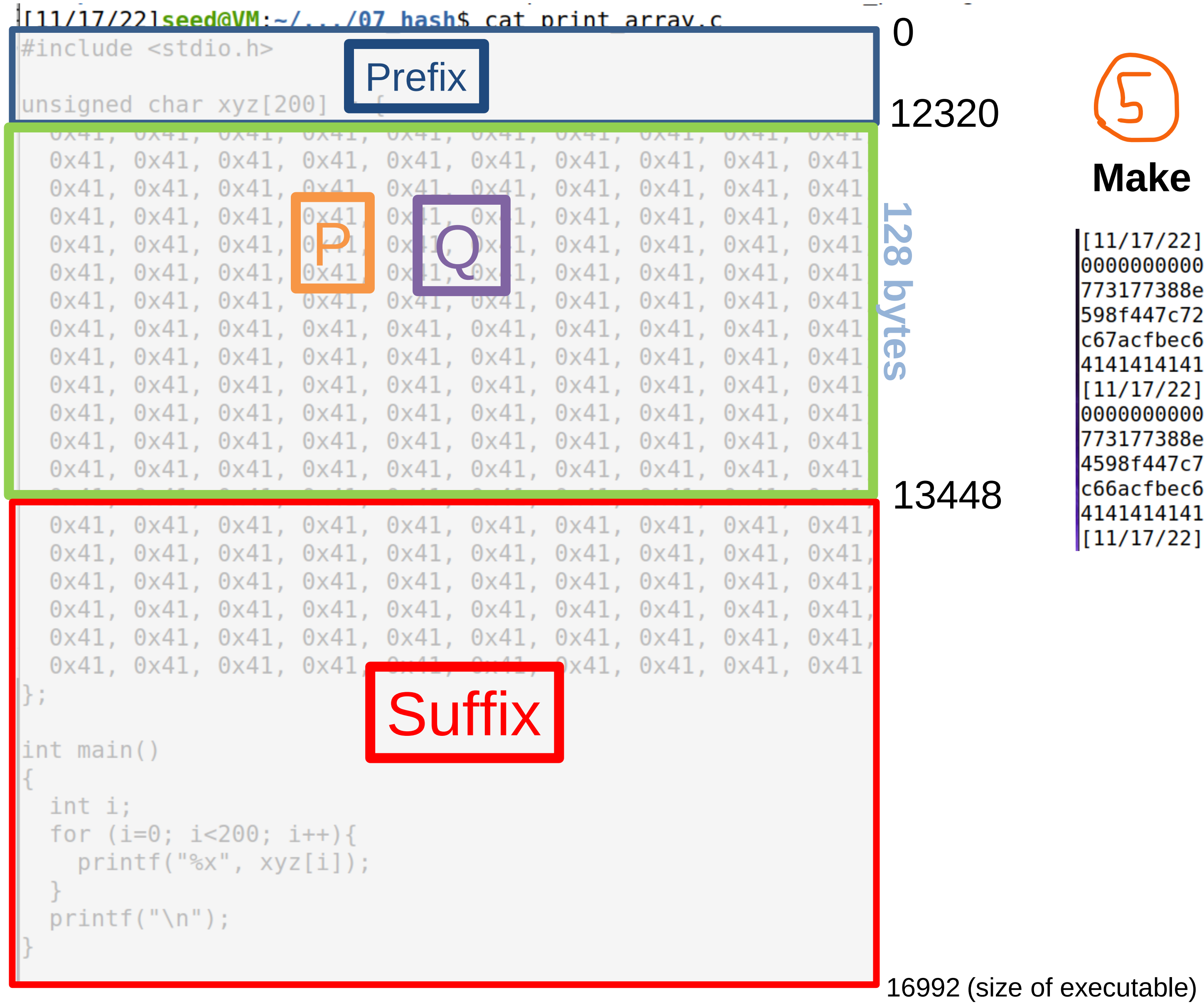
Verify that executables are different, but have the same hash



```
[11/17/22]seed@VM:~/hash_lab$ diff program1.out program2.out
Binary files program1.out and program2.out differ
[11/17/22]seed@VM:~/hash_lab$ md5sum program1.out
f489a326ed9c692f31eabccab06062ce  program1.out
[11/17/22]seed@VM:~/hash_lab$ md5sum program2.out
f489a326ed9c692f31eabccab06062ce  program2.out
```

✓

Task 4 on the lab

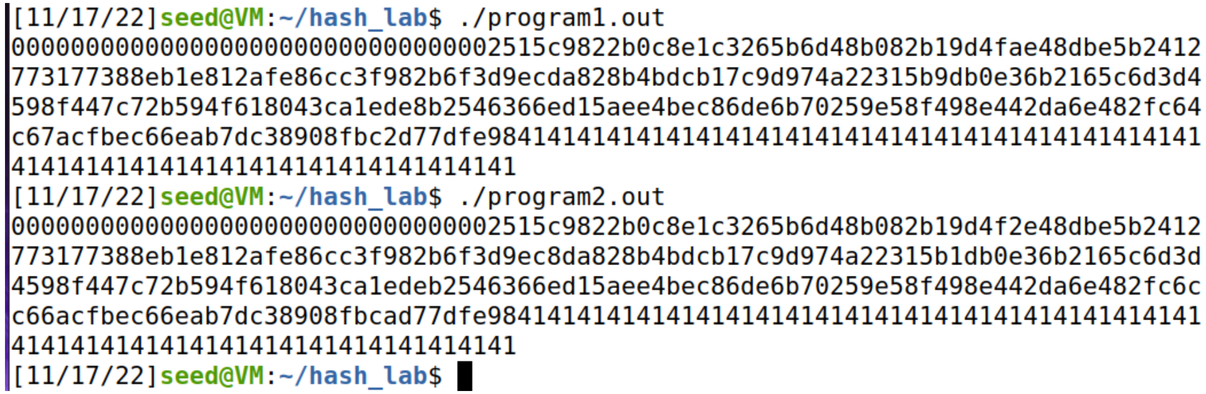


```
[11/17/22]seed@VM:~/.../07_hash$ cat print_array.c
#include <stdio.h>

unsigned char xyz[200] = {
  0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41
  0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41
  0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41
  0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41
  0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41
  0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41
  0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41
  0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41
  0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41
  0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41
  0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41
  0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41
  0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41,
  0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41,
  0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41,
  0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41,
  0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41,
  0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41
};

int main()
{
  int i;
  for (i=0; i<200; i++){
    printf("%x", xyz[i]);
  }
  printf("\n");
}
```

16992 (size of executable)

⑤

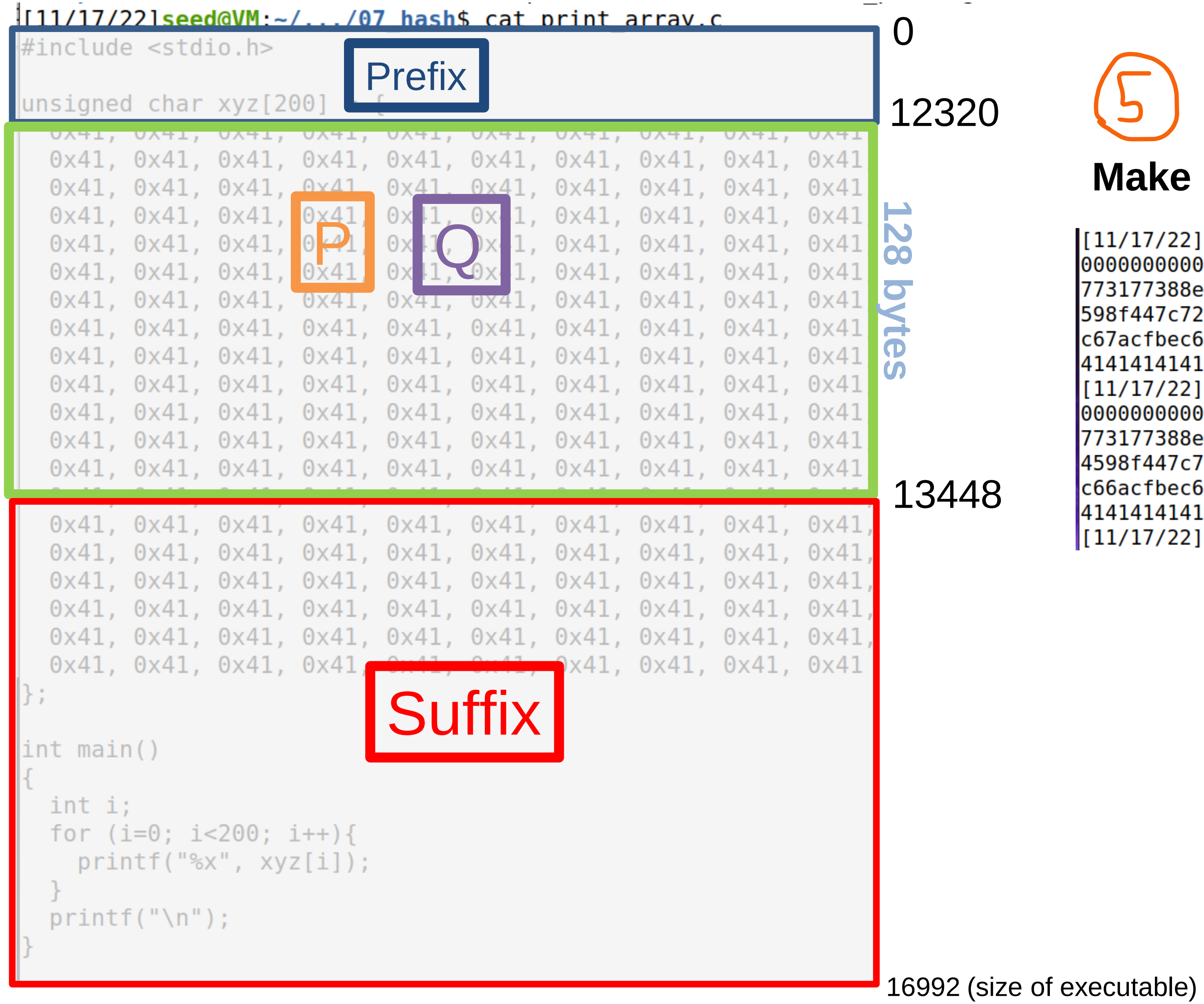
Make sure you still have a valid program ☺



```
[11/17/22]seed@VM:~/hash_lab$ ./program1.out
000000000000000000000000000000002515c9822b0c8e1c3265b6d48b082b19d4fae48dbe5b2412
773177388eb1e812afe86cc3f982b6f3d9ecda828b4bdcb17c9d974a22315b9db0e36b2165c6d3d4
598f447c72b594f618043ca1ede8b2546366ed15aee4bec86de6b70259e58f498e442da6e482fc64
c67acfbec66eab7dc38908fbc2d77dfe984141414141414141414141414141414141414141414141
4141414141414141414141414141414141
[11/17/22]seed@VM:~/hash_lab$ ./program2.out
000000000000000000000000000000002515c9822b0c8e1c3265b6d48b082b19d4f2e48dbe5b2412
773177388eb1e812afe86cc3f982b6f3d9ec8da828b4bdcb17c9d974a22315b1db0e36b2165c6d3d
4598f447c72b594f618043ca1edeb2546366ed15aee4bec86de6b70259e58f498e442da6e482fc6c
c66acfbec66eab7dc38908fbcad77dfe984141414141414141414141414141414141414141414141
4141414141414141414141414141414141
[11/17/22]seed@VM:~/hash_lab$ █
```

Somewhere in this output, you should find a small difference

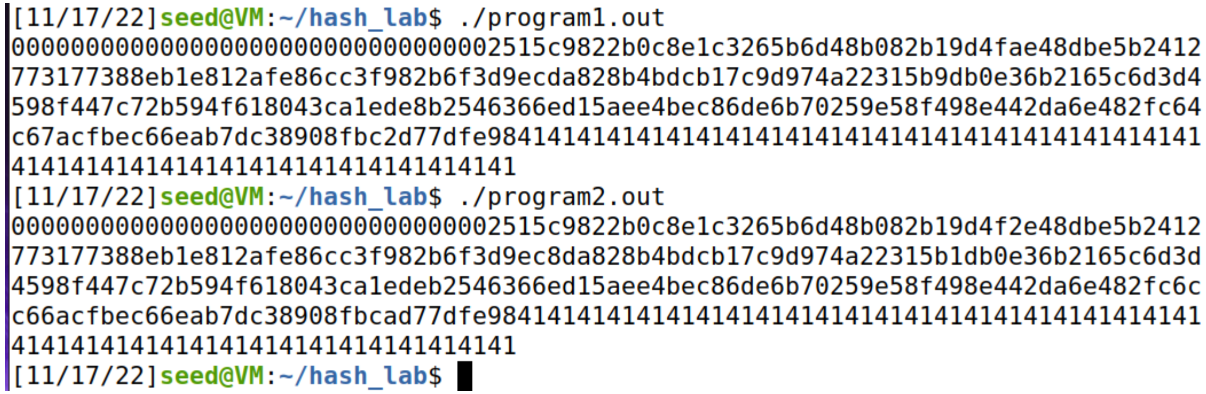
Task 4 on the lab



```
[11/17/22]seed@VM:~/.../07_hash$ cat print_array.c
#include <stdio.h>

unsigned char xyz[200] = {
  0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41
  0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41
  0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41
  0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41
  0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41
  0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41
  0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41
  0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41
  0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41
  0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41
  0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41
  0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41
  0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41,
  0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41,
  0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41,
  0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41,
  0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41,
  0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41
};

int main()
{
  int i;
  for (i=0; i<200; i++){
    printf("%x", xyz[i]);
  }
  printf("\n");
}
```

⑤

Make sure you still have a valid program ☺



```
[11/17/22]seed@VM:~/hash_lab$ ./program1.out
000000000000000000000000000000002515c9822b0c8e1c3265b6d48b082b19d4fae48dbe5b2412
773177388eb1e812afe86cc3f982b6f3d9ecda828b4bdcb17c9d974a22315b9db0e36b2165c6d3d4
598f447c72b594f618043ca1ede8b2546366ed15aee4bec86de6b70259e58f498e442da6e482fc64
c67acfbec66eab7dc38908fbc2d77dfe984141414141414141414141414141414141414141414141
4141414141414141414141414141414141
[11/17/22]seed@VM:~/hash_lab$ ./program2.out
000000000000000000000000000000002515c9822b0c8e1c3265b6d48b082b19d4f2e48dbe5b2412
773177388eb1e812afe86cc3f982b6f3d9ec8da828b4bdcb17c9d974a22315b1db0e36b2165c6d3d
4598f447c72b594f618043ca1edeb2546366ed15aee4bec86de6b70259e58f498e442da6e482fc6c
c66acfbec66eab7dc38908fbcad77dfe984141414141414141414141414141414141414141414141
4141414141414141414141414141414141
[11/17/22]seed@VM:~/hash_lab$ █
```

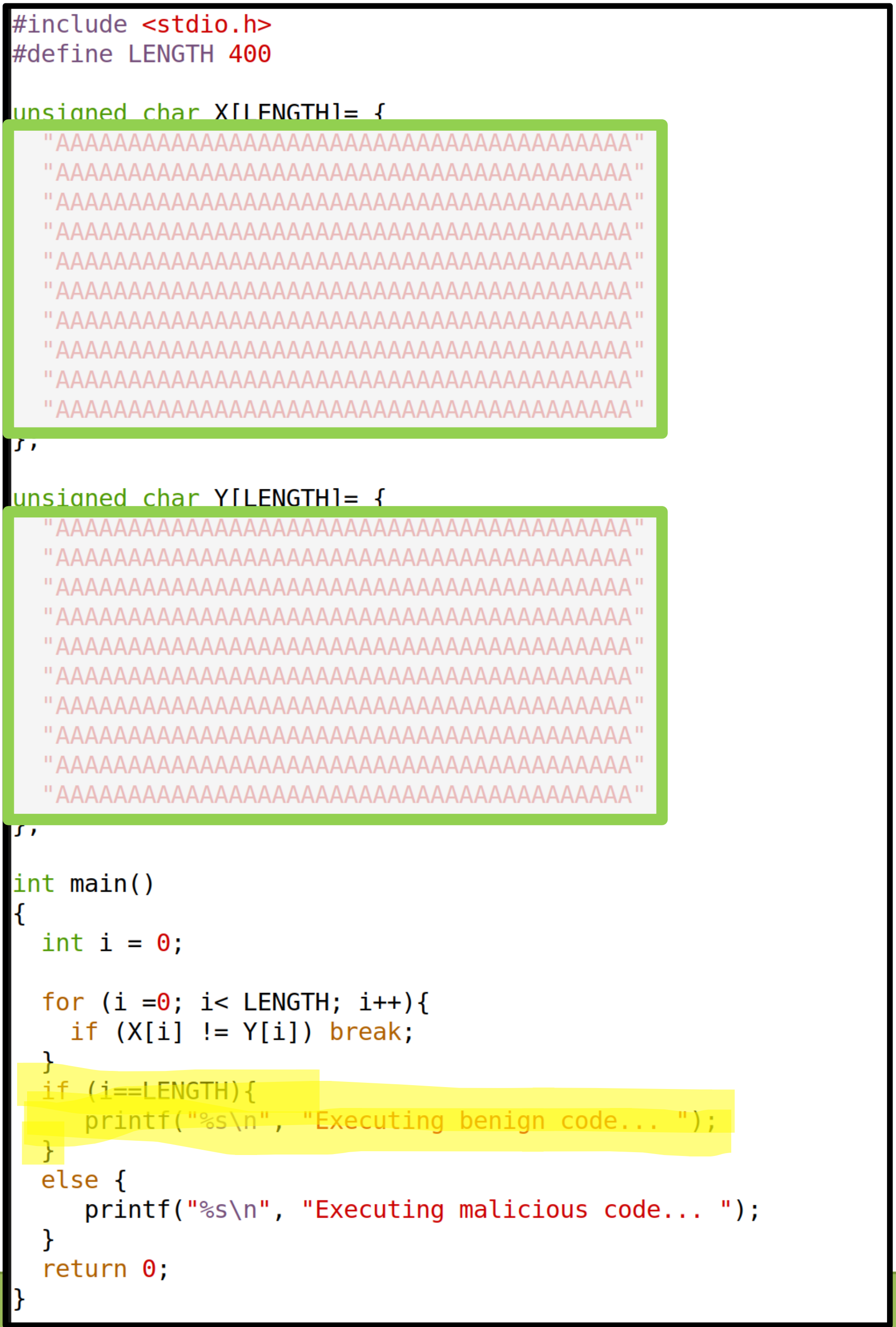
These programs print out different things, which is very benign

Our next goal is to write two programs with the same MD5 hash, but one does something malicious, and the other does something benign

```
#include <stdio.h>
#define LENGTH 400

unsigned char X[LENGTH]= {
  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
};

unsigned char Y[LENGTH]= {
  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
};

int main()
{
  int i = 0;

  for (i =0; i< LENGTH; i++){
    if (X[i] != Y[i]) break;
  }
  if (i==LENGTH){
     printf("%s\n", "Executing benign code... ");
  }
  else {
     printf("%s\n", "Executing malicious code... ");
  }
  return 0;
}
```

This program has two arrays X and Y

The program compares the contents of these two arrays



```
#include <stdio.h>
#define LENGTH 400

unsigned char X[LENGTH]= {
  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
};

unsigned char Y[LENGTH]= {
  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
};

int main()
{
  int i = 0;

  for (i =0; i< LENGTH; i++){
    if (X[i] != Y[i]) break;
  }
  if (i==LENGTH){
     printf("%s\n", "Executing benign code... ");
  }
  else {
     printf("%s\n", "Executing malicious code... ");
  }
  return 0;
}
```

This program has two arrays X and Y

The program compares the contents of these two arrays

If the two arrays are the same, then it will execute the benign code

```c
#include <stdio.h>
#define LENGTH 400

unsigned char X[LENGTH]= {
  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
};

unsigned char Y[LENGTH]= {
  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
};

int main()
{
  int i = 0;

  for (i =0; i< LENGTH; i++){
    if (X[i] != Y[i]) break;
  }
  if (i==LENGTH){
     printf("%s\n", "Executing benign code... ");
  }
  else {
     printf("%s\n", "Executing malicious code... ");
  }
  return 0;
}
```

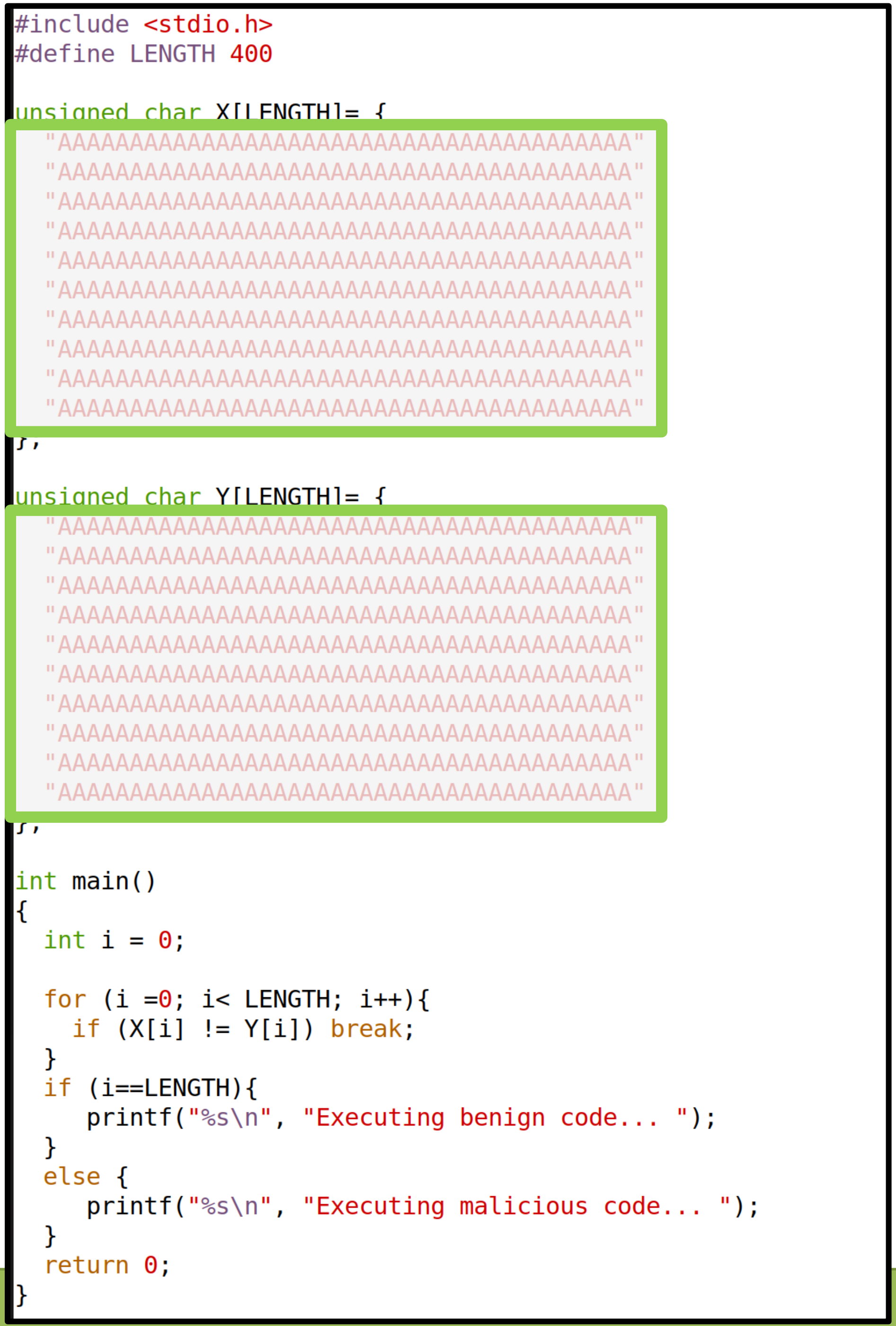
This program has two arrays X and Y

The program compares the contents of these two arrays

If the two arrays are the same, then it will execute the benign code

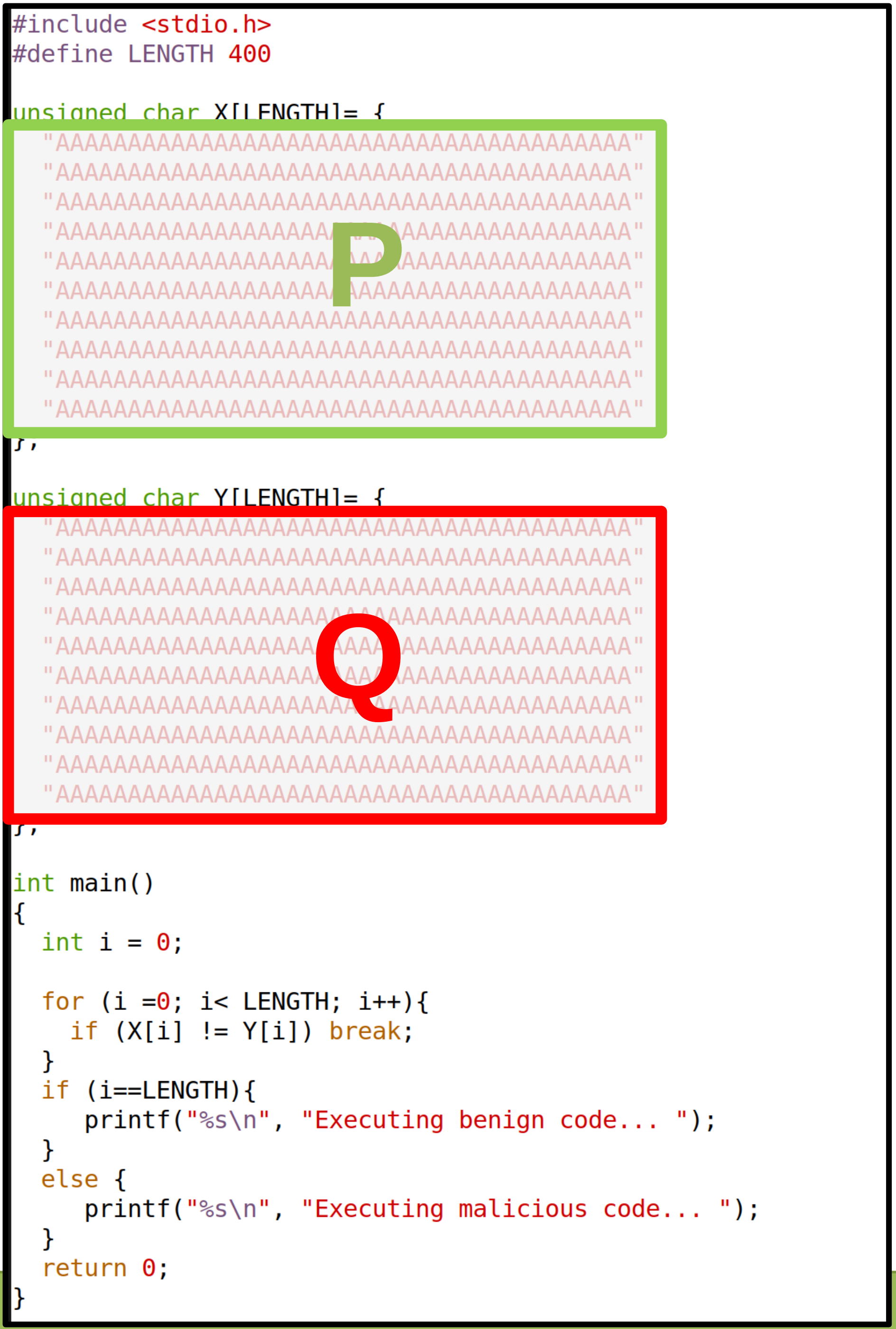
If the two arrays are different, then it will execute the benign code

```
#include <stdio.h>
#define LENGTH 400

unsigned char X[LENGTH]= {
  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
};

unsigned char Y[LENGTH]= {
  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
};

int main()
{
  int i = 0;

  for (i =0; i< LENGTH; i++){
    if (X[i] != Y[i]) break;
  }
  if (i==LENGTH){
     printf("%s\n", "Executing benign code... ");
  }
  else {
     printf("%s\n", "Executing malicious code... ");
  }
  return 0;
}
```

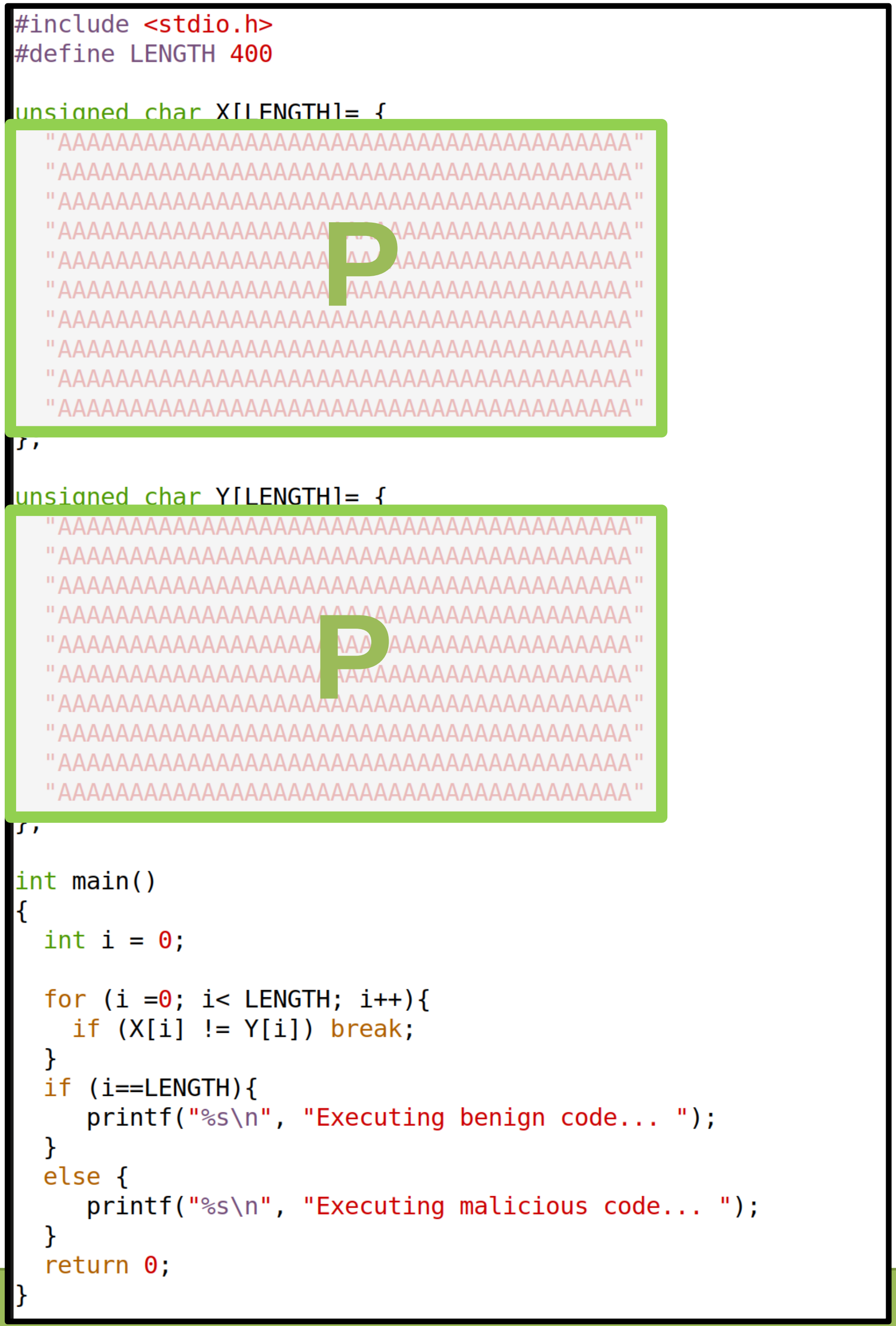


```
#include <stdio.h>
#define LENGTH 400

unsigned char X[LENGTH]= {
  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
};

unsigned char Y[LENGTH]= {
  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
};

int main()
{
  int i = 0;

  for (i =0; i< LENGTH; i++){
    if (X[i] != Y[i]) break;
  }
  if (i==LENGTH){
     printf("%s\n", "Executing benign code... ");
  }
  else {
     printf("%s\n", "Executing malicious code... ");
  }
  return 0;
}
```

Goal: Generate two variants of this program. One variant where the two arrays are the same (benign version), and one variant where the two arrays are different (malicious version)



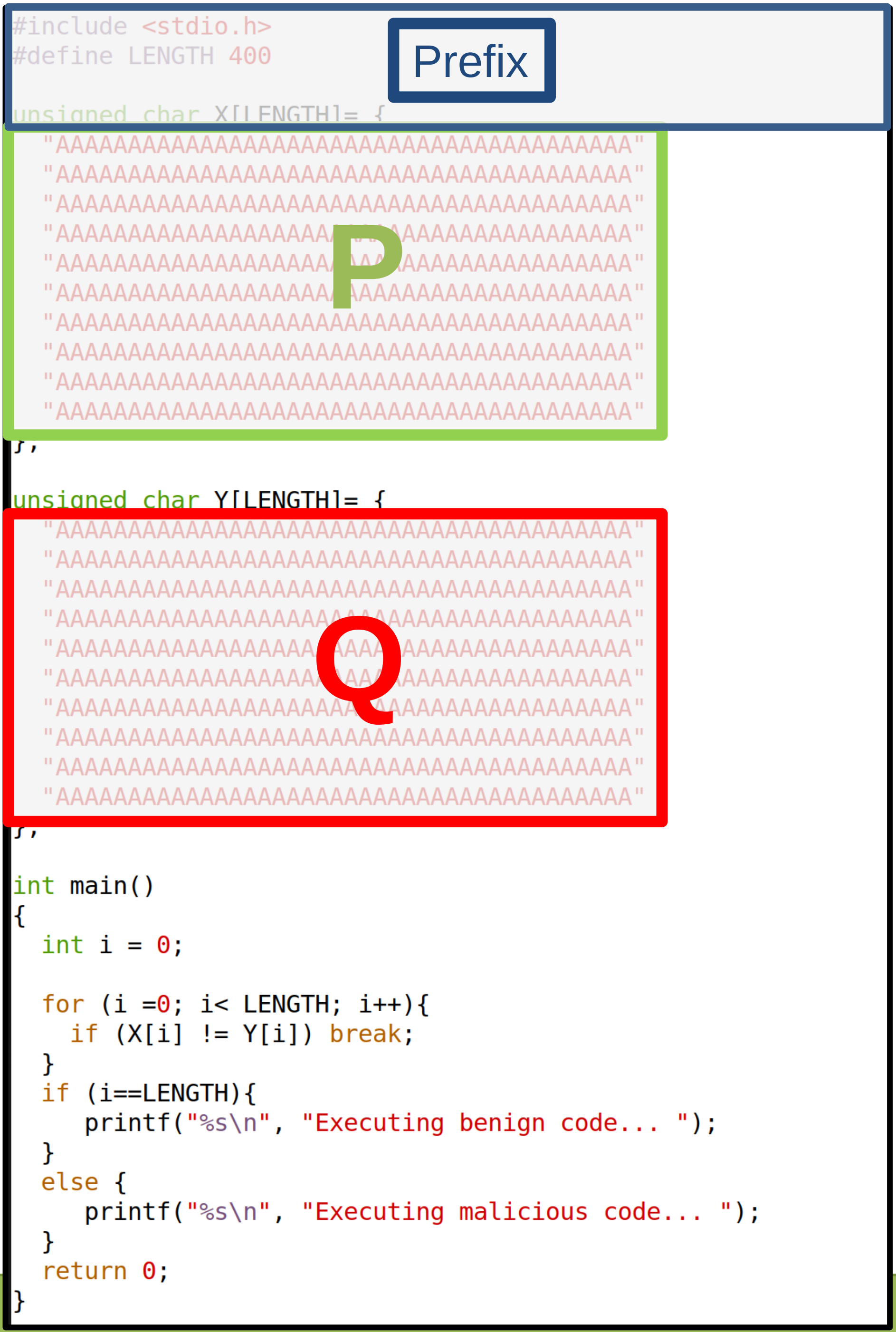
```
#include <stdio.h>
#define LENGTH 400

unsigned char X[LENGTH]= {
  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
};

unsigned char Y[LENGTH]= {
  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
};

int main()
{
  int i = 0;

  for (i =0; i< LENGTH; i++){
    if (X[i] != Y[i]) break;
  }
  if (i==LENGTH){
     printf("%s\n", "Executing benign code... ");
  }
  else {
     printf("%s\n", "Executing malicious code... ");
  }
  return 0;
}
```



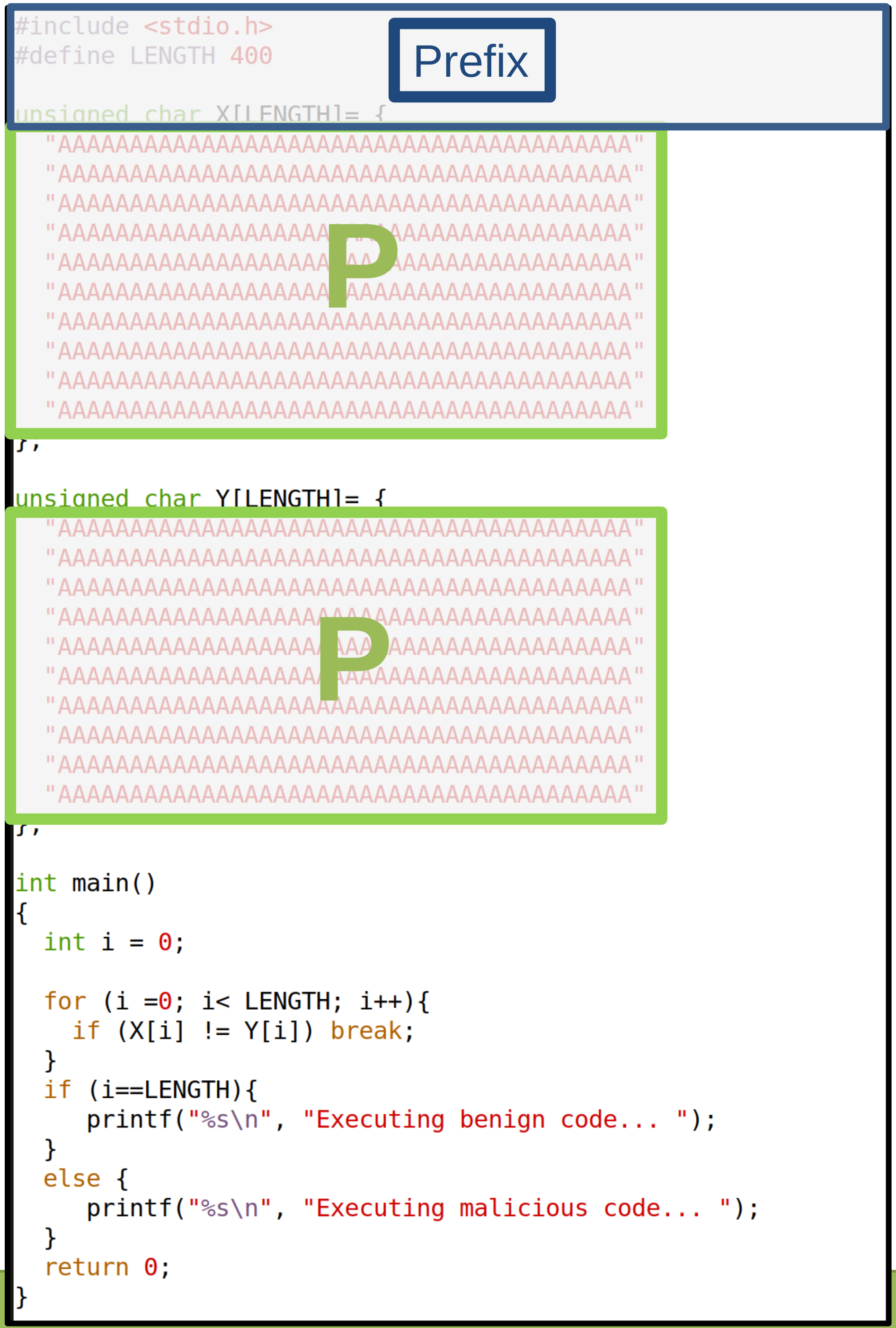
```
#include <stdio.h>
#define LENGTH 400

unsigned char X[LENGTH]= {
  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
};

unsigned char Y[LENGTH]= {
  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
};

int main()
{
  int i = 0;

  for (i =0; i< LENGTH; i++){
    if (X[i] != Y[i]) break;
  }
  if (i==LENGTH){
     printf("%s\n", "Executing benign code... ");
  }
  else {
     printf("%s\n", "Executing malicious code... ");
  }
  return 0;
}
```

Goal: Generate two variants of this program. One variant where the two arrays are the same (benign version), and one variant where the two arrays are different (malicious version)

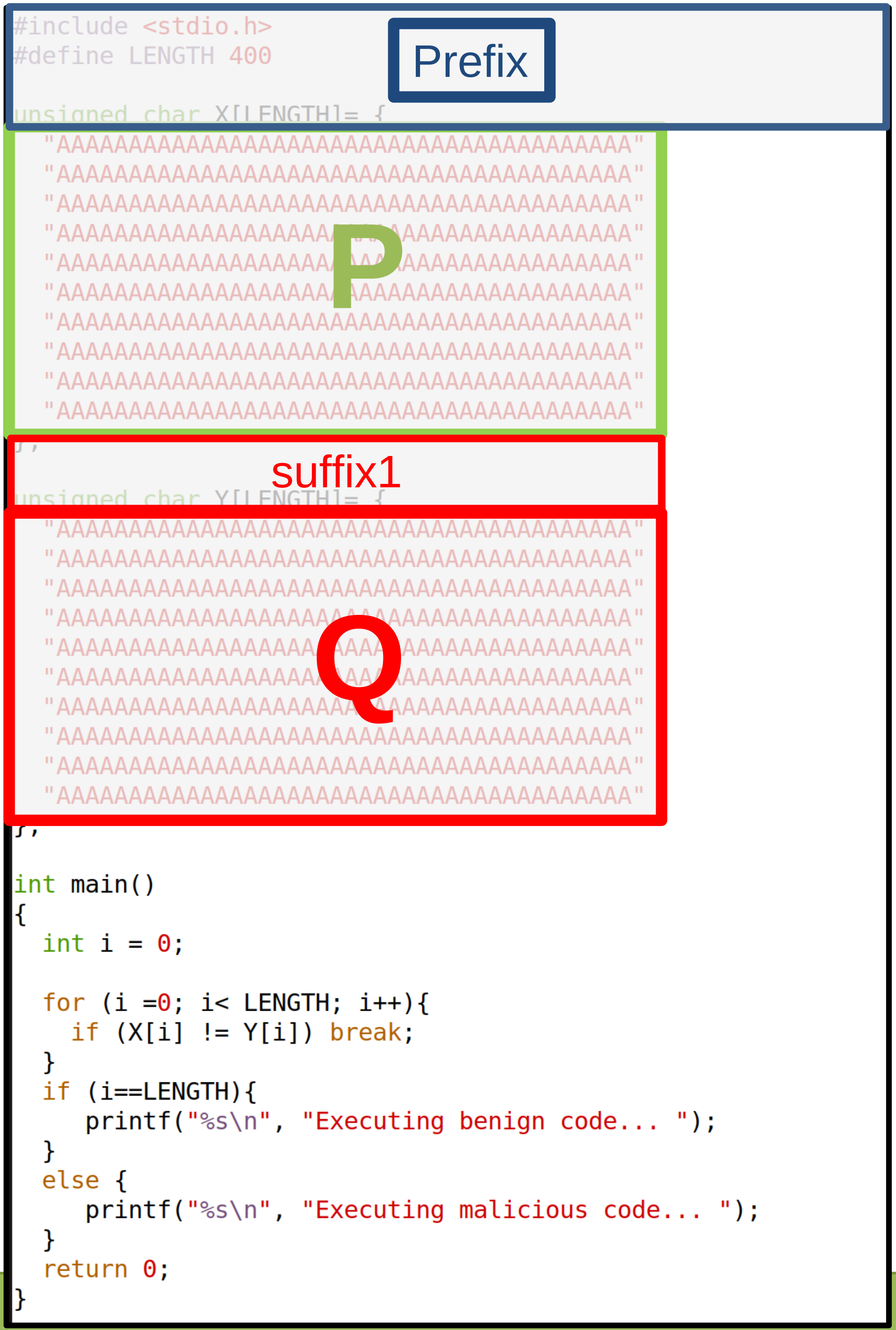
Because we can overwrite Array data without crashing the program, we will once again place **P** and **Q** (from md5collgen) into our arrays



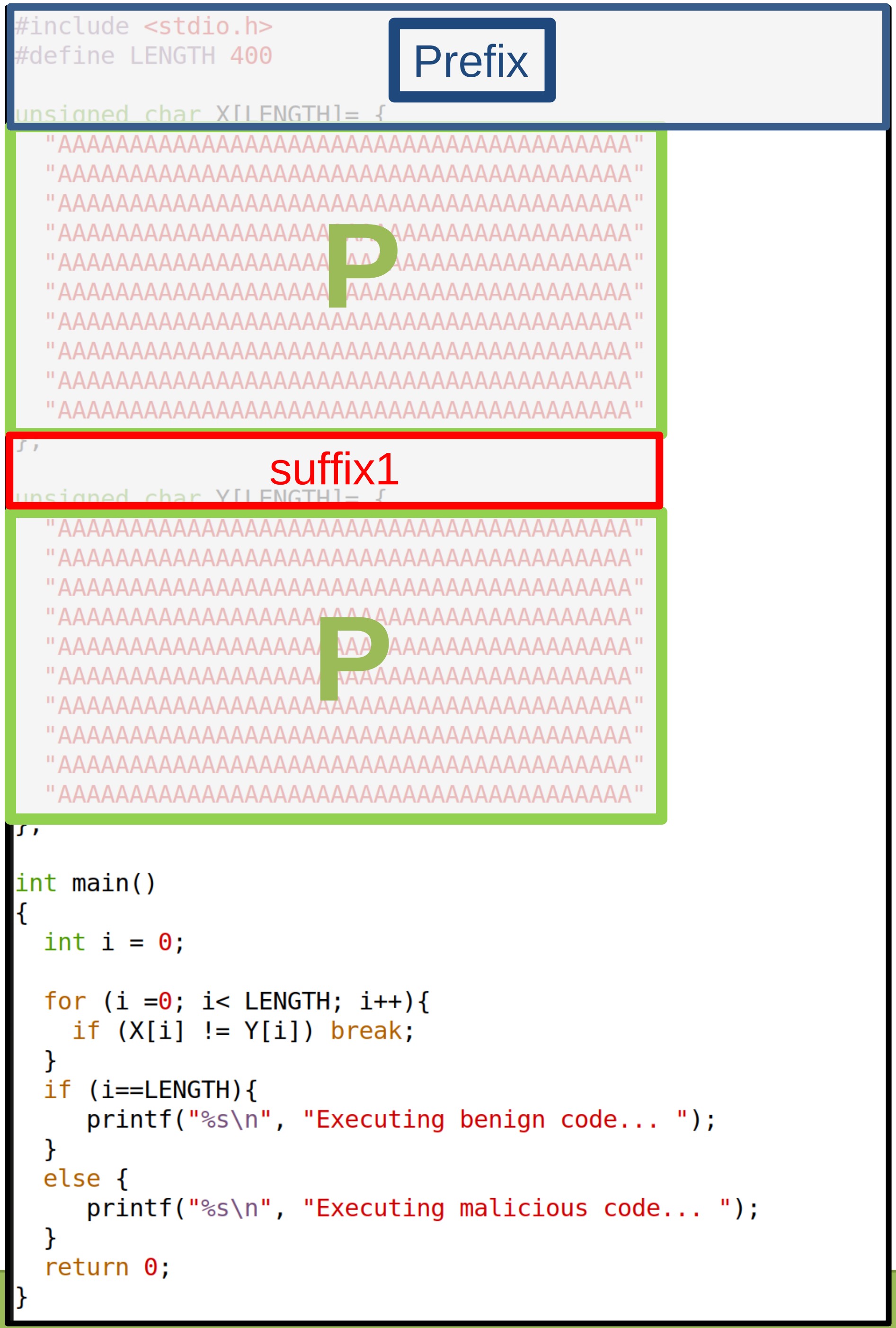
```
#include <stdio.h>
#define LENGTH 400

unsigned char X[LENGTH]= {
  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
};

unsigned char Y[LENGTH]= {
  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
};

int main()
{
  int i = 0;

  for (i =0; i< LENGTH; i++){
    if (X[i] != Y[i]) break;
  }
  if (i==LENGTH){
     printf("%s\n", "Executing benign code... ");
  }
  else {
     printf("%s\n", "Executing malicious code... ");
  }
  return 0;
}
```



```
#include <stdio.h>
#define LENGTH 400

unsigned char X[LENGTH]= {
  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
};

unsigned char Y[LENGTH]= {
  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
};

int main()
{
  int i = 0;

  for (i =0; i< LENGTH; i++){
    if (X[i] != Y[i]) break;
  }
  if (i==LENGTH){
     printf("%s\n", "Executing benign code... ");
  }
  else {
     printf("%s\n", "Executing malicious code... ");
  }
  return 0;
}
```

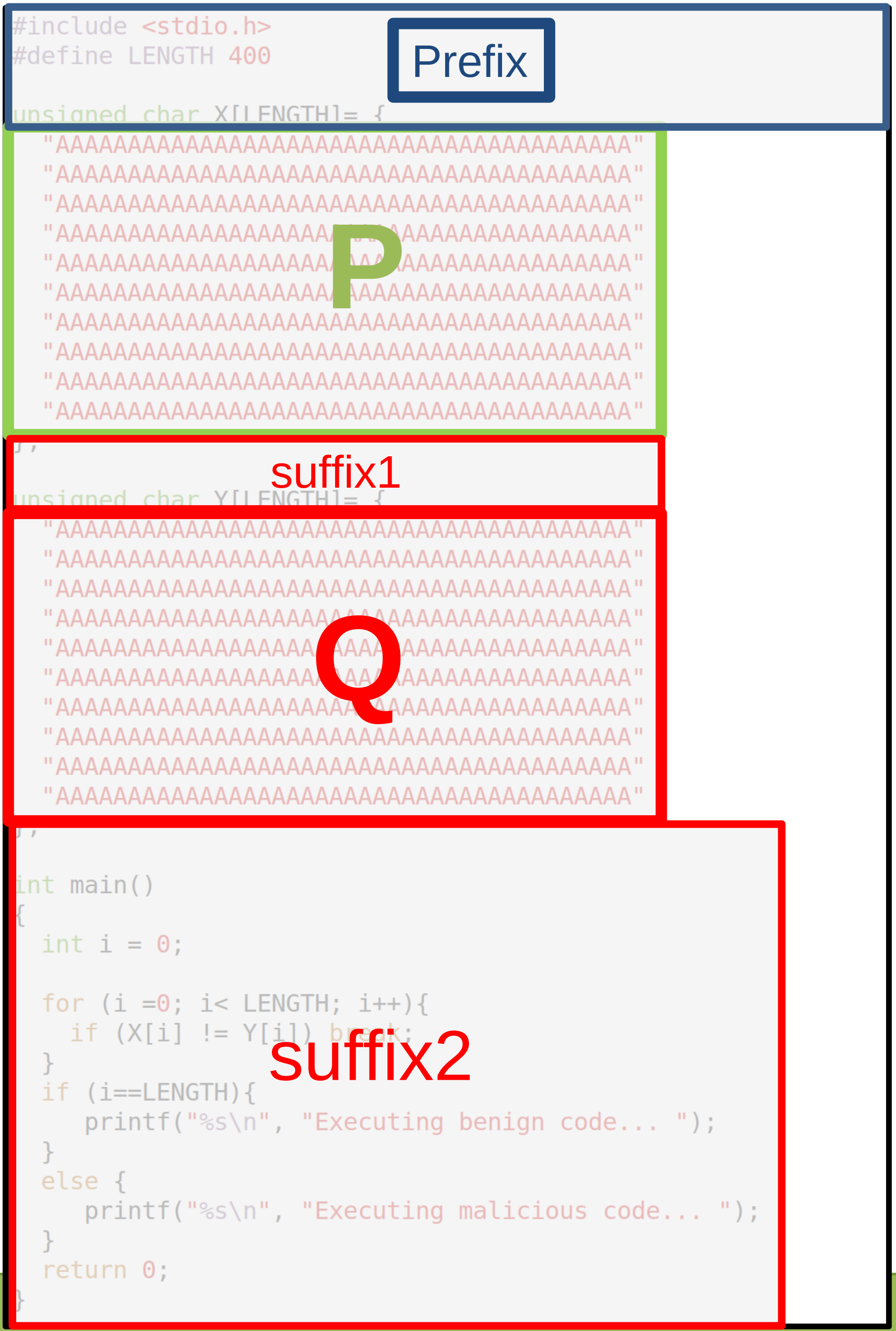


```c
#include <stdio.h>
#define LENGTH 400

unsigned char X[LENGTH]= {
  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
};

unsigned char Y[LENGTH]= {
  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
};

int main()
{
  int i = 0;

  for (i =0; i< LENGTH; i++){
    if (X[i] != Y[i]) break;
  }
  if (i==LENGTH){
     printf("%s\n", "Executing benign code... ");
  }
  else {
     printf("%s\n", "Executing malicious code... ");
  }
  return 0;
}
```

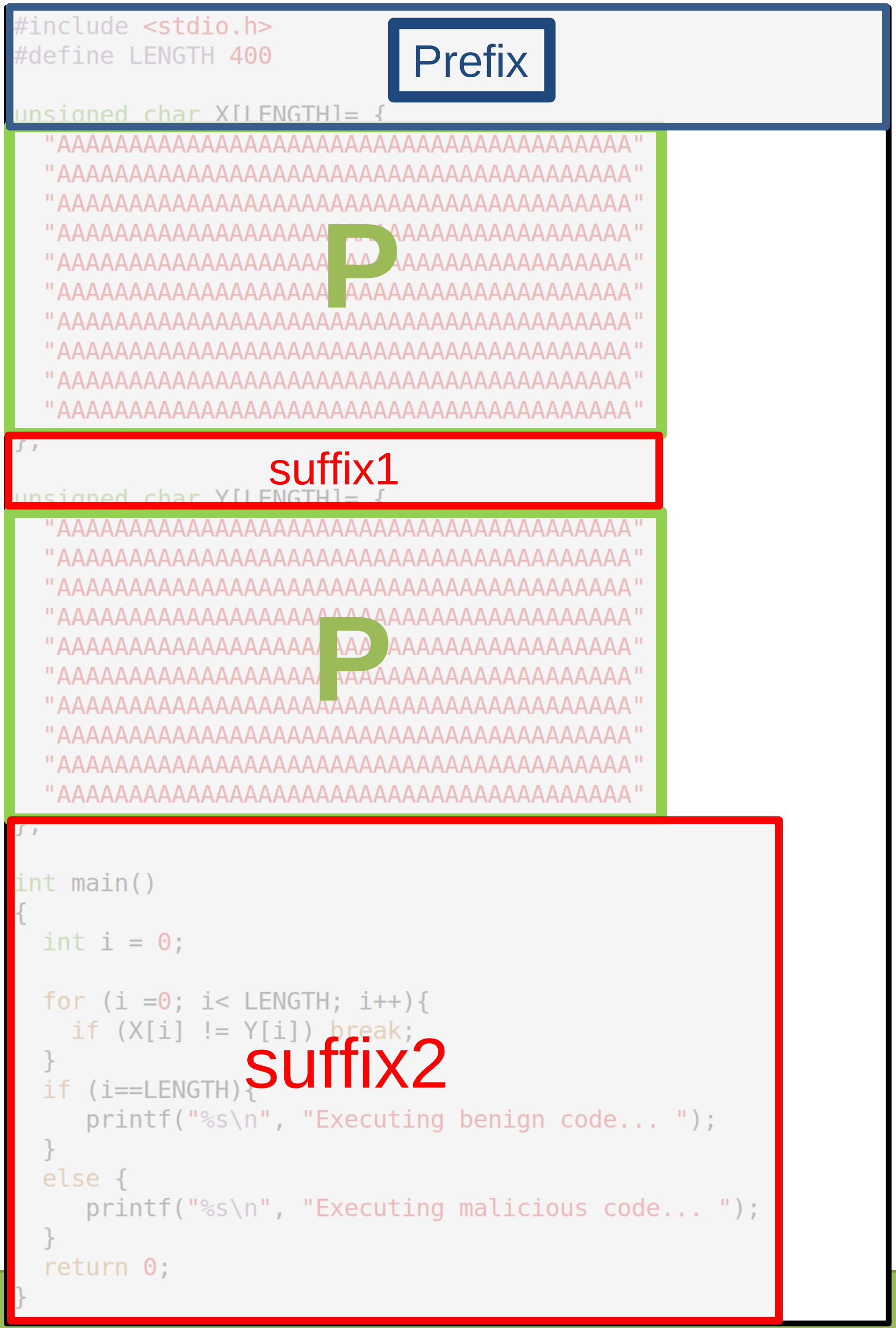


```c
#include <stdio.h>
#define LENGTH 400

unsigned char X[LENGTH]= {
  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
};

unsigned char Y[LENGTH]= {
  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
};

int main()
{
  int i = 0;

  for (i =0; i< LENGTH; i++){
    if (X[i] != Y[i]) break;
  }
  if (i==LENGTH){
     printf("%s\n", "Executing benign code... ");
  }
  else {
     printf("%s\n", "Executing malicious code... ");
  }
  return 0;
}
```

Because we are inserting P/Q at two different points into our program, we will have two suffixes



```
#include <stdio.h>
#define LENGTH 400

unsigned char X[LENGTH]= {
  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
};

unsigned char Y[LENGTH]= {
  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
};

int main()
{
  int i = 0;

  for (i =0; i< LENGTH; i++){
    if (X[i] != Y[i]) break;
  }
  if (i==LENGTH){
     printf("%s\n", "Executing benign code... ");
  }
  else {
     printf("%s\n", "Executing malicious code... ");
  }
  return 0;
}
```

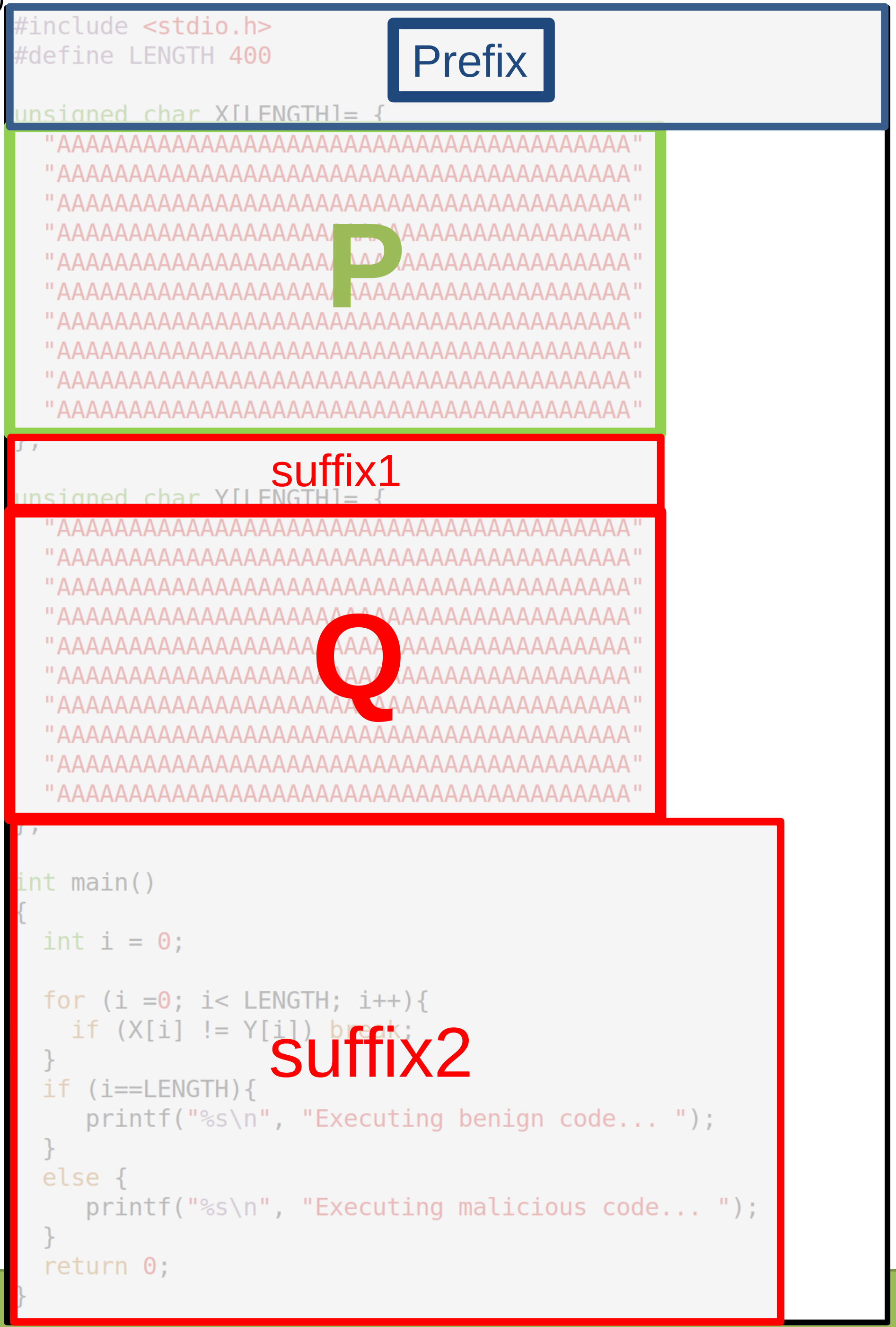


```
#include <stdio.h>
#define LENGTH 400

unsigned char X[LENGTH]= {
  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
};

unsigned char Y[LENGTH]= {
  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
};

int main()
{
  int i = 0;

  for (i =0; i< LENGTH; i++){
    if (X[i] != Y[i]) break;
  }
  if (i==LENGTH){
     printf("%s\n", "Executing benign code... ");
  }
  else {
     printf("%s\n", "Executing malicious code... ");
  }
  return 0;
}
```

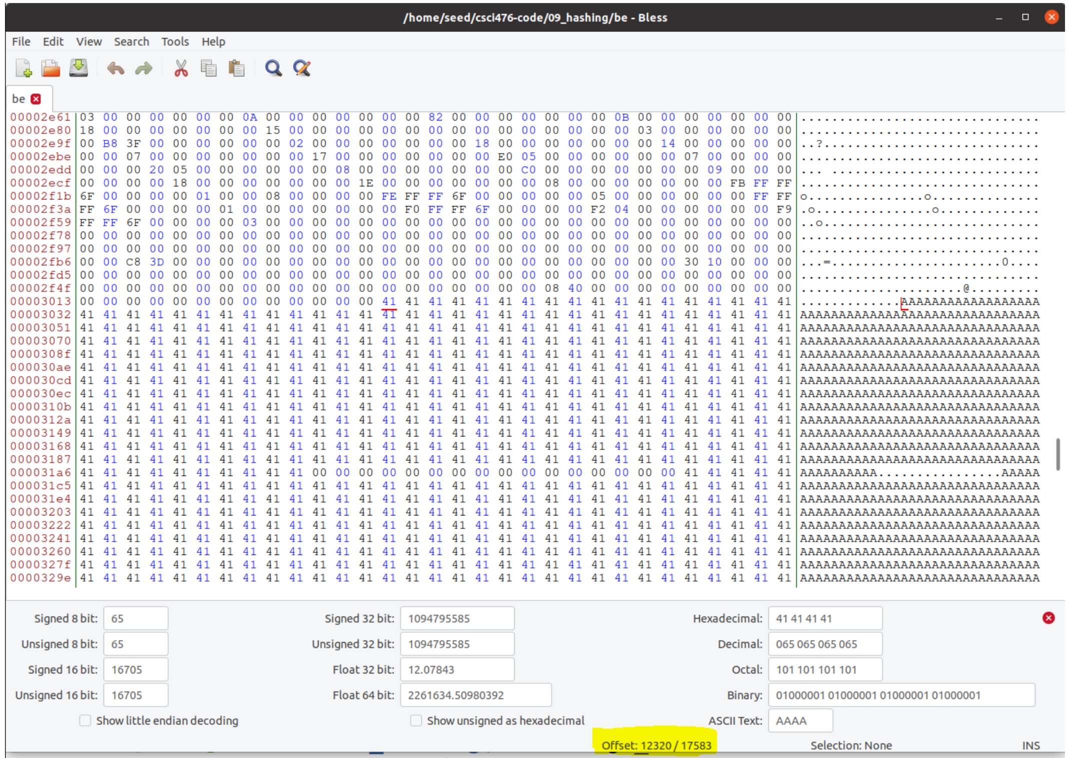
Because we are inserting P/Q at two different points into our program, we will have two suffixes

First, we must get the starting location of our array X

Memory Locations (Bytes)



```
#include <stdio.h>
#define LENGTH 400

unsigned char X[LENGTH]= {
  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
};

unsigned char Y[LENGTH]= {
  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
};

int main()
{
  int i = 0;

  for (i =0; i< LENGTH; i++){
    if (X[i] != Y[i]) break;
  }
  if (i==LENGTH){
     printf("%s\n", "Executing benign code... ");
  }
  else {
     printf("%s\n", "Executing malicious code... ");
  }
  return 0;
}
```

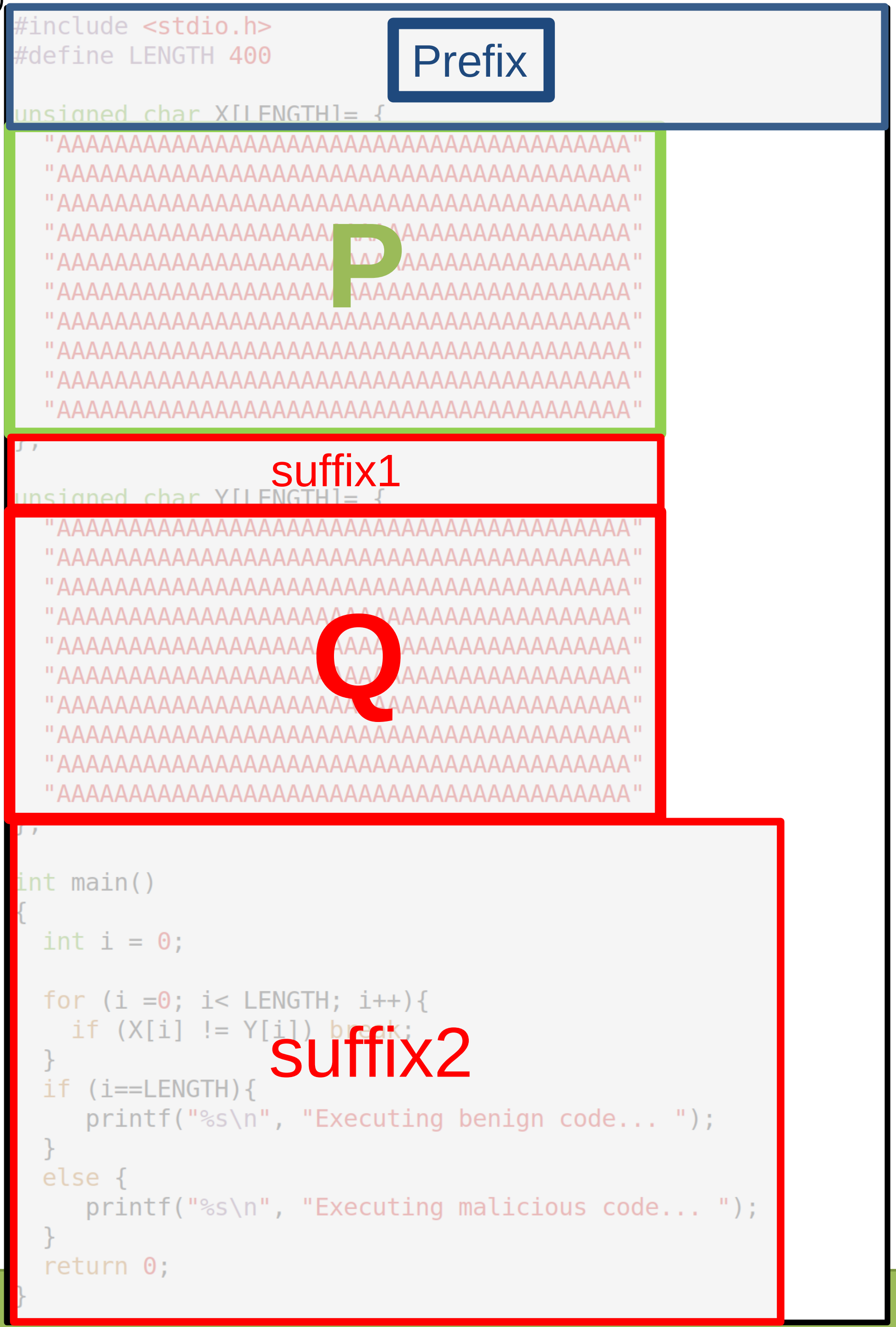


```
00002e61 03 00 00 00 00 00 00 0A 00 00 00 00 00 00 00 82 00 00 00 00 00 00 00 0B 00 00 00 00 00 00 00 ...............................
00002e80 18 00 00 00 00 00 00 00 15 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 03 00 00 00 00 00 00 ...............................
00002e9f 00 B8 3F 00 00 00 00 00 00 02 00 00 00 00 00 00 00 18 00 00 00 00 00 00 00 14 00 00 00 00 00 ..?............................
00002ebe 00 00 07 00 00 00 00 00 00 00 17 00 00 00 00 00 00 00 E0 05 00 00 00 00 00 00 07 00 00 00 00 ...............................
00002edd 00 00 00 20 05 00 00 00 00 00 00 08 00 00 00 00 00 00 00 C0 00 00 00 00 00 00 00 09 00 00 00 ... ...........................
00002ecf 00 00 00 00 18 00 00 00 00 00 00 00 1E 00 00 00 00 00 00 00 08 00 00 00 00 00 00 00 FB FF FF ...............................
00002f1b 6F 00 00 00 00 01 00 00 08 00 00 00 00 FE FF FF 6F 00 00 00 00 00 05 00 00 00 00 00 00 FF FF o...............o..............
00002f3a FF 6F 00 00 00 00 01 00 00 00 00 00 00 00 F0 FF FF 6F 00 00 00 00 F2 04 00 00 00 00 00 00 F9 .o...............o.............
00002f59 FF FF 6F 00 00 00 00 03 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ..o............................
00002f78 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ...............................
00002f97 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ...............................
00002fb6 00 00 C8 3D 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 30 10 00 00 00 ...=......................0....
00002fd5 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ...............................
00002f4f 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 08 40 00 00 00 00 00 00 00 00 00 .....................@.........
00003013 00 00 00 00 00 00 00 00 00 00 00 00 00 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 .............AAAAAAAAAAAAAAAAAA
00003032 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
00003051 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
00003070 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
0000308f 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
000030ae 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
000030cd 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
000030ec 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
0000310b 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
0000312a 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
00003149 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
00003168 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
00003187 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
000031a6 41 41 41 41 41 41 41 41 41 41 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 41 41 41 41 41 AAAAAAAAAA................AAAAA
000031c5 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
000031e4 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
00003203 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
00003222 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
00003241 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
00003260 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
0000327f 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
0000329e 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
```

Signed 8 bit: 65
Unsigned 8 bit: 65
Signed 16 bit: 16705
Unsigned 16 bit: 16705
Signed 32 bit: 1094795585
Unsigned 32 bit: 1094795585
Float 32 bit: 12.07843
Float 64 bit: 2261634.50980392
Hexadecimal: 41 41 41 41
Decimal: 065 065 065 065
Octal: 101 101 101 101
Binary: 01000001 01000001 01000001 01000001
ASCII Text: AAAA
☐ Show little endian decoding
☐ Show unsigned as hexadecimal
Offset: 12320 / 17583
Selection: None
INS

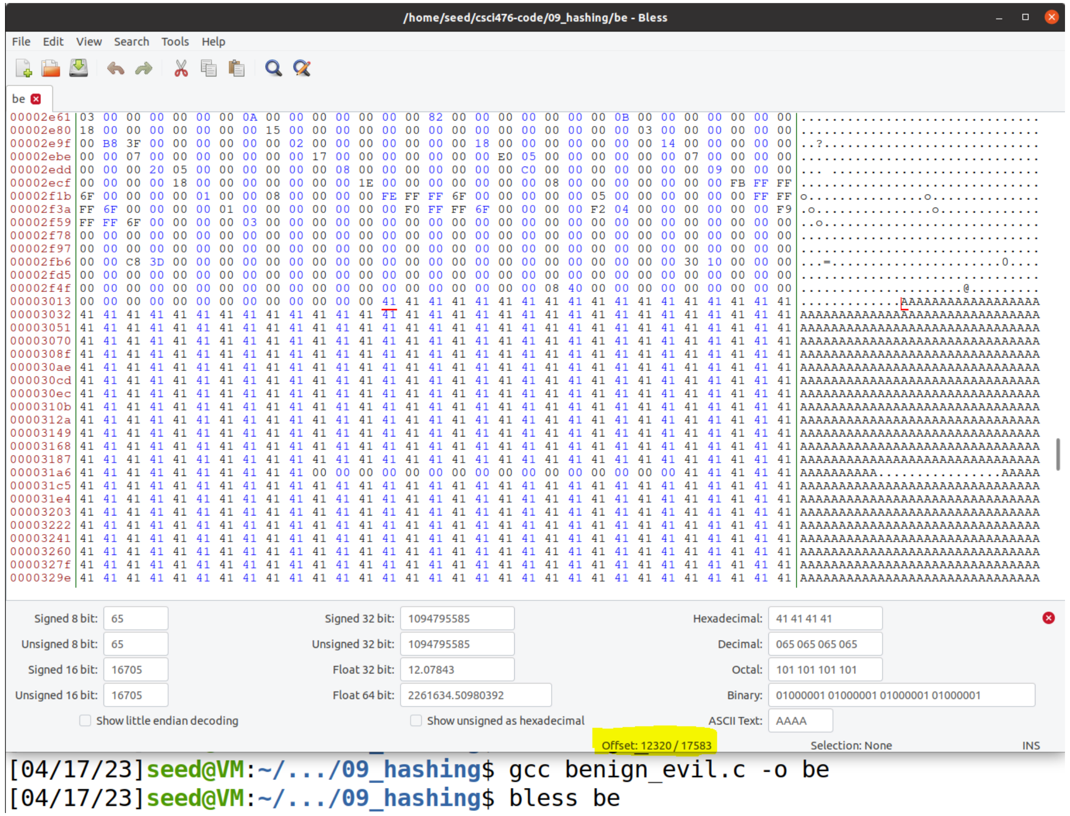
```
[04/17/23]seed@VM:~/.../09_hashing$ gcc benign_evil.c -o be
[04/17/23]seed@VM:~/.../09_hashing$ bless be
```

Memory Locations (Bytes)



```c
#include <stdio.h>
#define LENGTH 400

unsigned char X[LENGTH]= {
  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
};

unsigned char Y[LENGTH]= {
  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
};

int main()
{
  int i = 0;

  for (i =0; i< LENGTH; i++){
    if (X[i] != Y[i]) break;
  }
  if (i==LENGTH){
     printf("%s\n", "Executing benign code... ");
  }
  else {
     printf("%s\n", "Executing malicious code... ");
  }
  return 0;
}
```

First, we must get the starting location of our array X



```
00002e61 03 00 00 00 00 00 00 0A 00 00 00 00 00 00 00 82 00 00 00 00 00 00 00 0B 00 00 00 00 00 00 00 ...............................
00002e80 18 00 00 00 00 00 00 00 15 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 03 00 00 00 00 00 00 ...............................
00002e9f 00 B8 3F 00 00 00 00 00 00 02 00 00 00 00 00 00 00 18 00 00 00 00 00 00 00 14 00 00 00 00 00 ..?............................
00002ebe 00 00 07 00 00 00 00 00 00 00 17 00 00 00 00 00 00 00 E0 05 00 00 00 00 00 00 07 00 00 00 00 ...............................
00002edd 00 00 00 20 05 00 00 00 00 00 00 08 00 00 00 00 00 00 00 C0 00 00 00 00 00 00 00 09 00 00 00 ... ...........................
00002ecf 00 00 00 00 18 00 00 00 00 00 00 00 1E 00 00 00 00 00 00 00 08 00 00 00 00 00 00 00 FB FF FF ...............................
00002f1b 6F 00 00 00 00 01 00 00 08 00 00 00 00 FE FF FF 6F 00 00 00 00 00 05 00 00 00 00 00 00 FF FF o...............o..............
00002f3a FF 6F 00 00 00 00 01 00 00 00 00 00 00 00 F0 FF FF 6F 00 00 00 00 F2 04 00 00 00 00 00 00 F9 .o...............o.............
00002f59 FF FF 6F 00 00 00 00 03 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ..o............................
00002f78 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ...............................
00002f97 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ...............................
00002fb6 00 00 C8 3D 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 30 10 00 00 00 ...=......................0....
00002fd5 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ...............................
00002f4f 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 08 40 00 00 00 00 00 00 00 00 00 .....................@.........
00003013 00 00 00 00 00 00 00 00 00 00 00 00 00 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 .............AAAAAAAAAAAAAAAAAA
00003032 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
00003051 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
00003070 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
0000308f 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
000030ae 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
000030cd 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
000030ec 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
0000310b 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
0000312a 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
00003149 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
00003168 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
00003187 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
000031a6 41 41 41 41 41 41 41 41 41 41 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 41 41 41 41 41 AAAAAAAAAA................AAAAA
000031c5 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
000031e4 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
00003203 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
00003222 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
00003241 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
00003260 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
0000327f 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
0000329e 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
```

Signed 8 bit: 65
Unsigned 8 bit: 65
Signed 16 bit: 16705
Unsigned 16 bit: 16705
☐ Show little endian decoding

Signed 32 bit: 1094795585
Unsigned 32 bit: 1094795585
Float 32 bit: 12.07843
Float 64 bit: 2261634.50980392
☐ Show unsigned as hexadecimal

Hexadecimal: 41 41 41 41
Decimal: 065 065 065 065
Octal: 101 101 101 101
Binary: 01000001 01000001 01000001 01000001
ASCII Text: AAAA

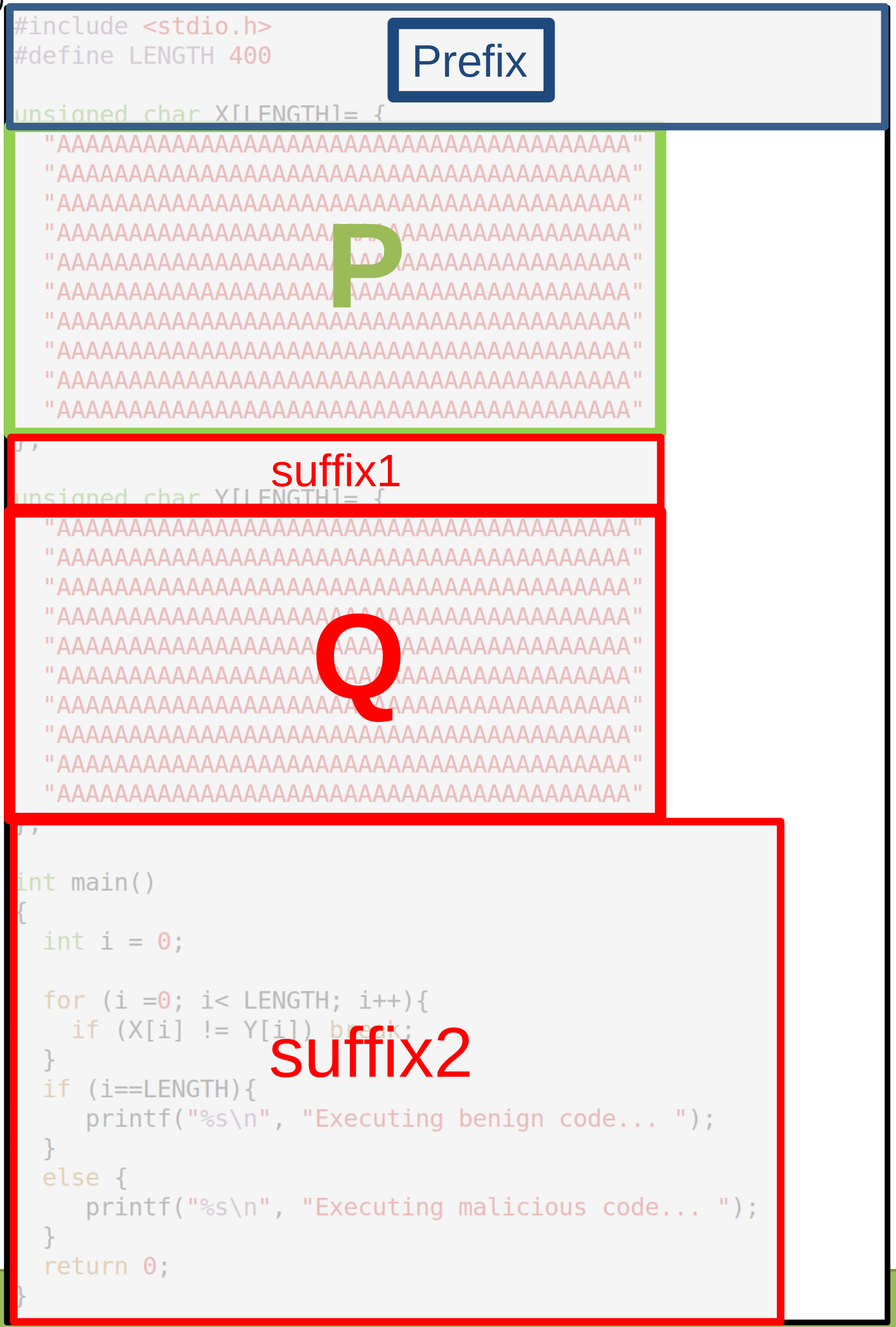
Offset: 12320 / 17583   Selection: None   INS

```
[04/17/23]seed@VM:~/.../09_hashing$ gcc benign_evil.c -o be
[04/17/23]seed@VM:~/.../09_hashing$ bless be
```

Offset = 12320

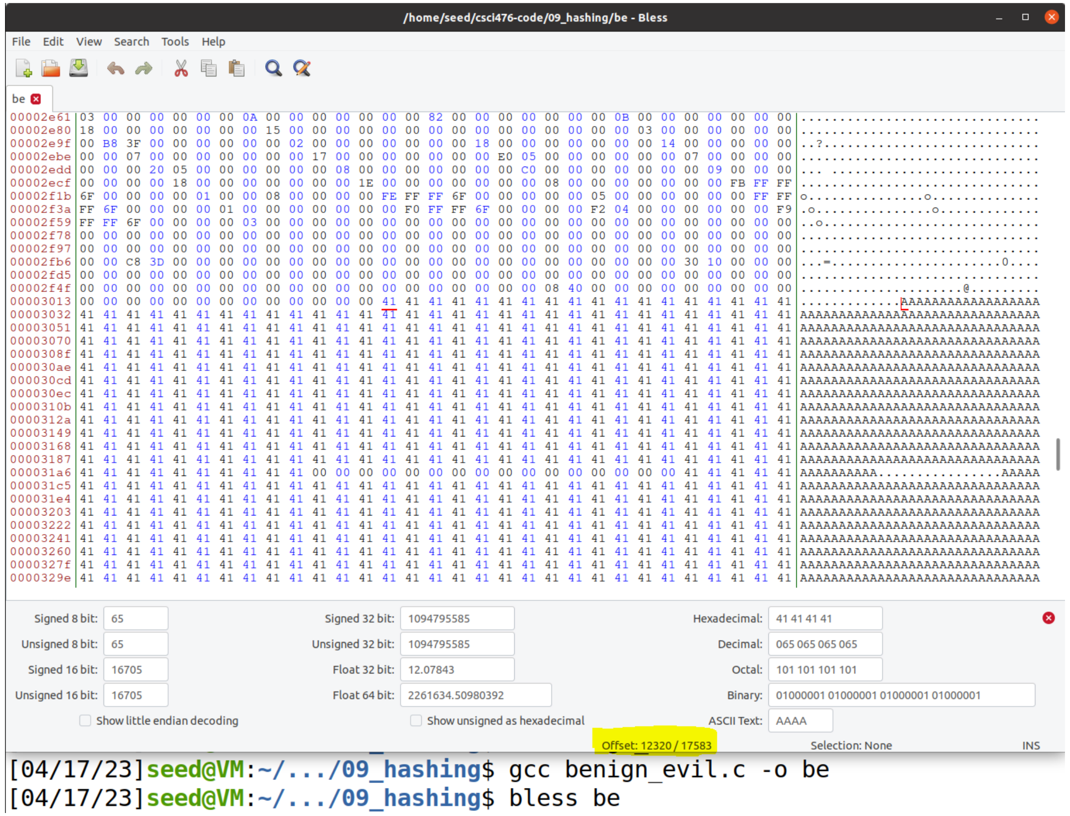
Padding will mess up out attack, so we must make sure that padding doesn't get added

Offset = 12352 (multiple of 64)



```
#include <stdio.h>
#define LENGTH 400

unsigned char X[LENGTH]= {
  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
};

unsigned char Y[LENGTH]= {
  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
};

int main()
{
  int i = 0;

  for (i =0; i< LENGTH; i++){
    if (X[i] != Y[i]) break;
  }
  if (i==LENGTH){
    printf("%s\n", "Executing benign code... ");
  }
  else {
    printf("%s\n", "Executing malicious code... ");
  }
  return 0;
}
```

First, we must get the starting location of our array X



```
00002e61 03 00 00 00 00 00 00 0A 00 00 00 00 00 00 00 82 00 00 00 00 00 00 00 0B 00 00 00 00 00 00 00 ...............................
00002e80 18 00 00 00 00 00 00 00 15 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 03 00 00 00 00 00 00 ...............................
00002e9f 00 B8 3F 00 00 00 00 00 00 02 00 00 00 00 00 00 00 18 00 00 00 00 00 00 00 14 00 00 00 00 00 ..?............................
00002ebe 00 00 07 00 00 00 00 00 00 00 17 00 00 00 00 00 00 00 E0 05 00 00 00 00 00 00 07 00 00 00 00 ...............................
00002edd 00 00 00 20 05 00 00 00 00 00 00 08 00 00 00 00 00 00 00 C0 00 00 00 00 00 00 00 09 00 00 00 ... ...........................
00002ecf 00 00 00 00 18 00 00 00 00 00 00 00 1E 00 00 00 00 00 00 00 08 00 00 00 00 00 00 00 FB FF FF ...............................
00002f1b 6F 00 00 00 00 01 00 00 08 00 00 00 00 FE FF FF 6F 00 00 00 00 00 05 00 00 00 00 00 00 FF FF o...............o..............
00002f3a FF 6F 00 00 00 00 01 00 00 00 00 00 00 00 F0 FF FF 6F 00 00 00 00 F2 04 00 00 00 00 00 00 F9 .o...............o.............
00002f59 FF FF 6F 00 00 00 00 03 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ..o............................
00002f78 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ...............................
00002f97 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ...............................
00002fb6 00 00 C8 3D 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 30 10 00 00 00 ...=......................0....
00002fd5 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ...............................
00002f4f 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 08 40 00 00 00 00 00 00 00 00 00 .....................@.........
00003013 00 00 00 00 00 00 00 00 00 00 00 00 00 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 .............AAAAAAAAAAAAAAAAAA
00003032 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
00003051 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
00003070 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
0000308f 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
000030ae 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
000030cd 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
000030ec 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
0000310b 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
0000312a 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
00003149 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
00003168 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
00003187 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
000031a6 41 41 41 41 41 41 41 41 41 41 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 41 41 41 41 41 AAAAAAAAAA................AAAAA
000031c5 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
000031e4 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
00003203 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
00003222 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
00003241 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
00003260 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
0000327f 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
0000329e 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
```

Signed 8 bit: 65
Unsigned 8 bit: 65
Signed 16 bit: 16705
Unsigned 16 bit: 16705
☐ Show little endian decoding
Signed 32 bit: 1094795585
Unsigned 32 bit: 1094795585
Float 32 bit: 12.07843
Float 64 bit: 2261634.50980392
☐ Show unsigned as hexadecimal
Hexadecimal: 41 41 41 41
Decimal: 065 065 065 065
Octal: 101 101 101 101
Binary: 01000001 01000001 01000001 01000001
ASCII Text: AAAA

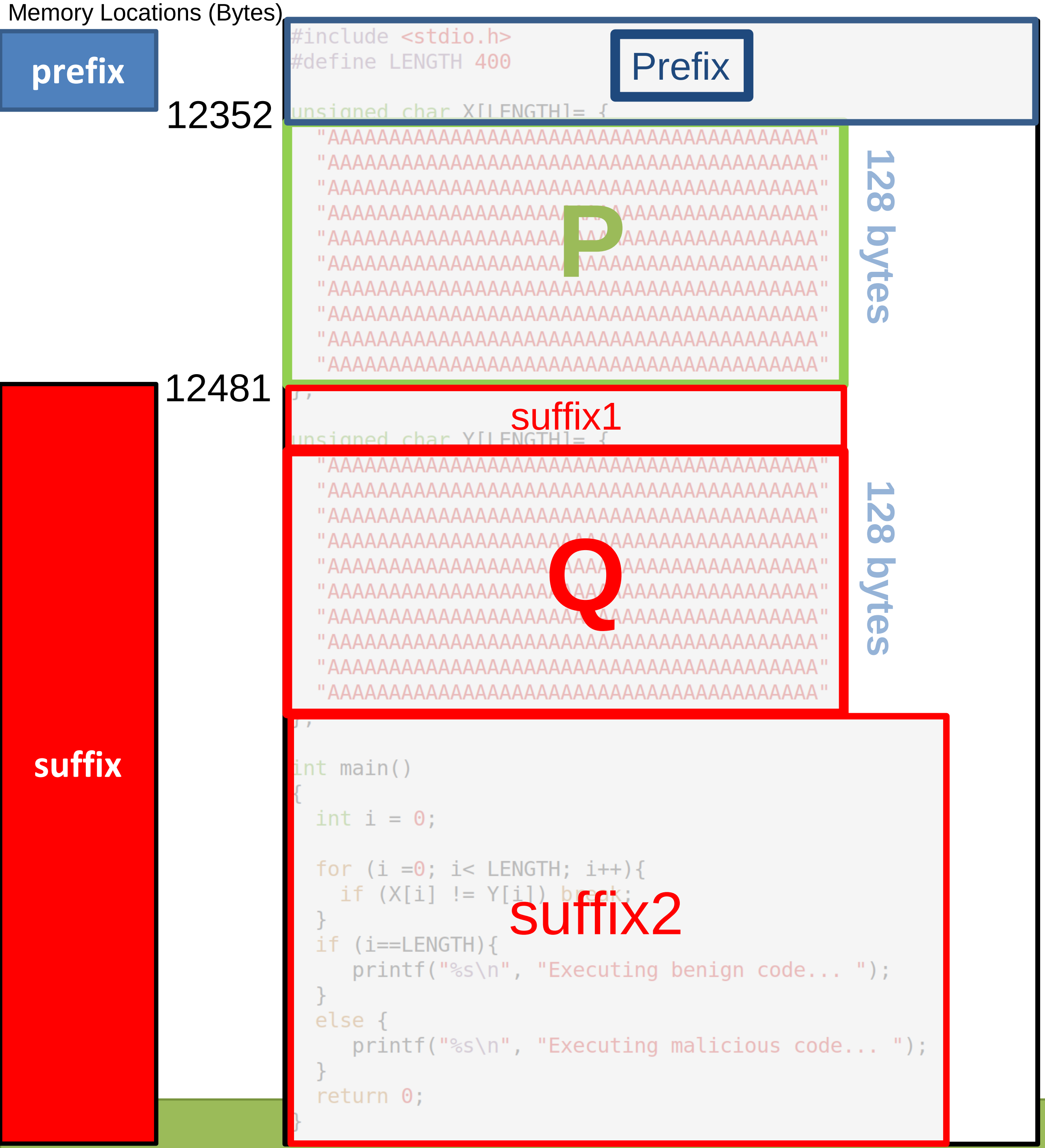
Offset: 12320 / 17583 Selection: None INS

```
[04/17/23]seed@VM:~/.../09_hashing$ gcc benign_evil.c -o be
[04/17/23]seed@VM:~/.../09_hashing$ bless be
```

Offset = 12320

Padding will mess up out attack, so we must make sure that padding doesn't get added

Offset = 12352 (multiple of 64)



```
#include <stdio.h>
#define LENGTH 400

unsigned char X[LENGTH]= {
  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
};

unsigned char Y[LENGTH]= {
  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
};

int main()
{
  int i = 0;

  for (i =0; i< LENGTH; i++){
    if (X[i] != Y[i]) break;
  }
  if (i==LENGTH){
     printf("%s\n", "Executing benign code... ");
  }
  else {
     printf("%s\n", "Executing malicious code... ");
  }
  return 0;
}
```

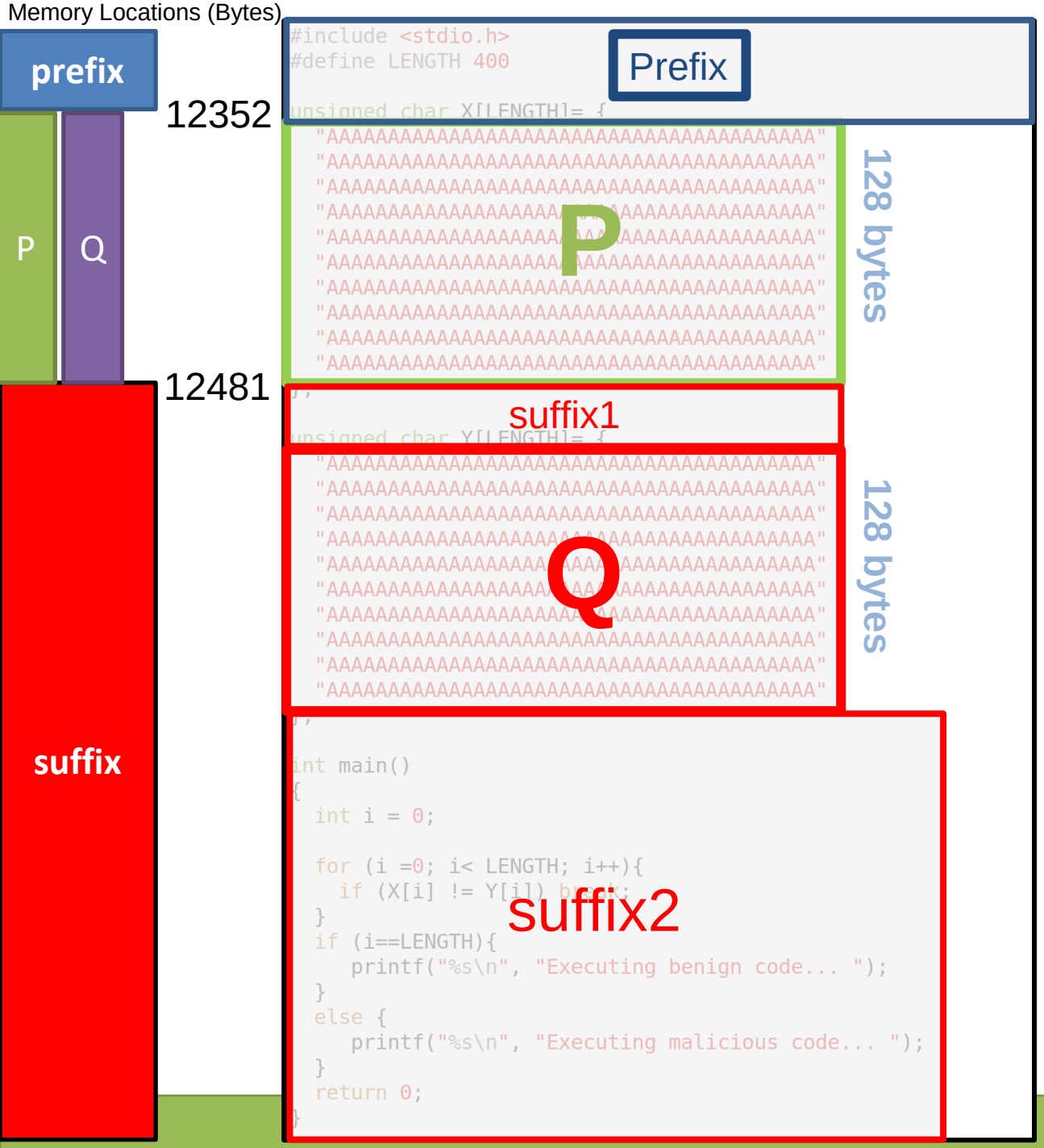
Offset = 12352

P and Q will once again be 128 bytes long

Therefore, P ends at (12352 + 128) = 12480

(we have to add +1 when getting the suffix to prevent getting an extra byte)

```
[04/17/23]seed@VM:~/.../09_hashing$ head -c 12352 be > prefix
[04/17/23]seed@VM:~/.../09_hashing$ tail -c +12481 be > suffix
```

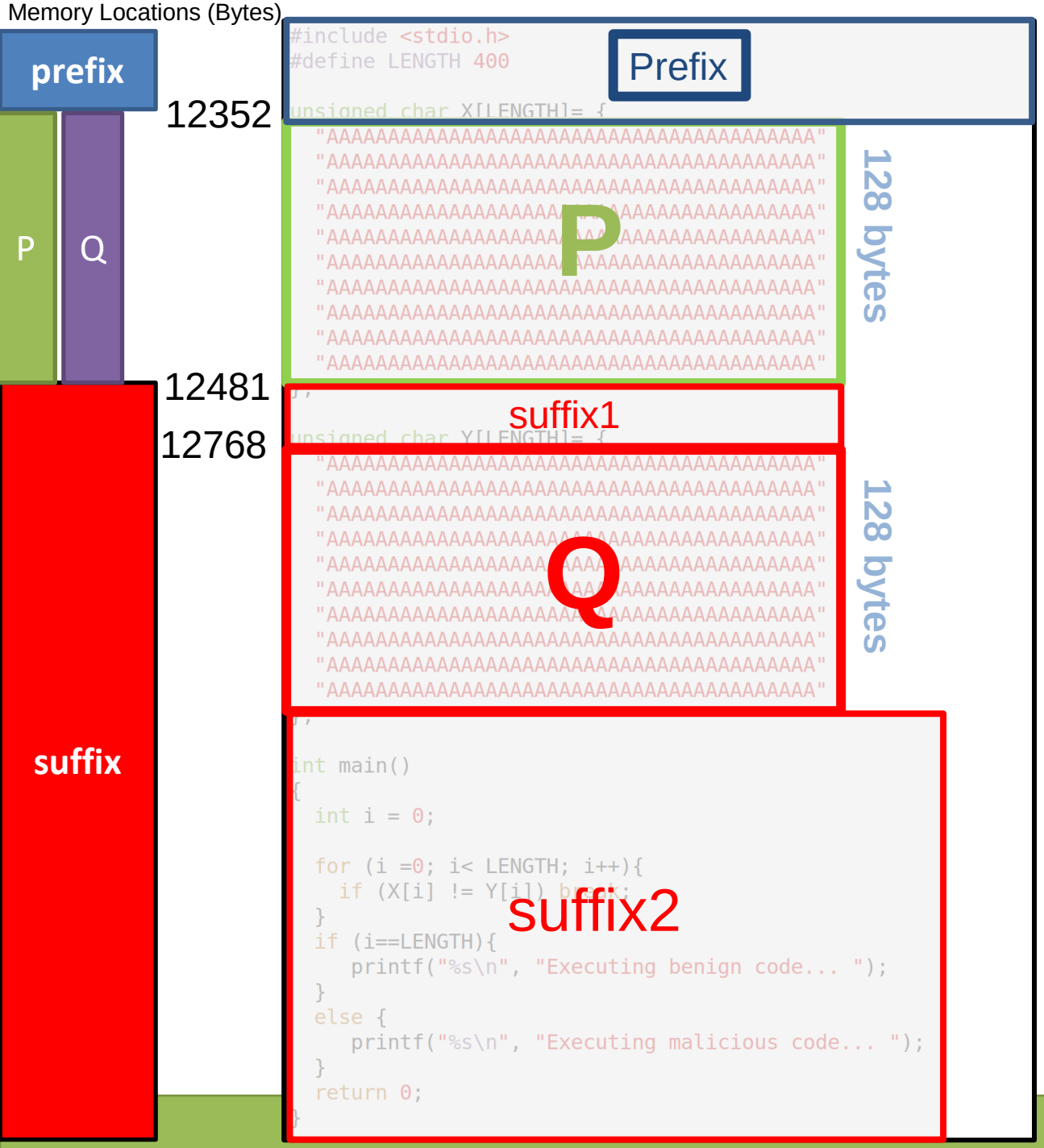


Now that we have the prefix, we can generate P and Q

```
[04/17/23]seed@VM:~/.../09_hashing$ md5collgen -p prefix -o out1 out2
MD5 collision generator v1.5
by Marc Stevens (http://www.win.tue.nl/hashclash/)

Using output filenames: 'out1' and 'out2'
Using prefixfile: 'prefix'
Using initial value: fe5a2a34aa864ea9110d0e2fa43e0327

Generating first block: ........
Generating second block: S10.........................................
Running time: 4.60817 s
```

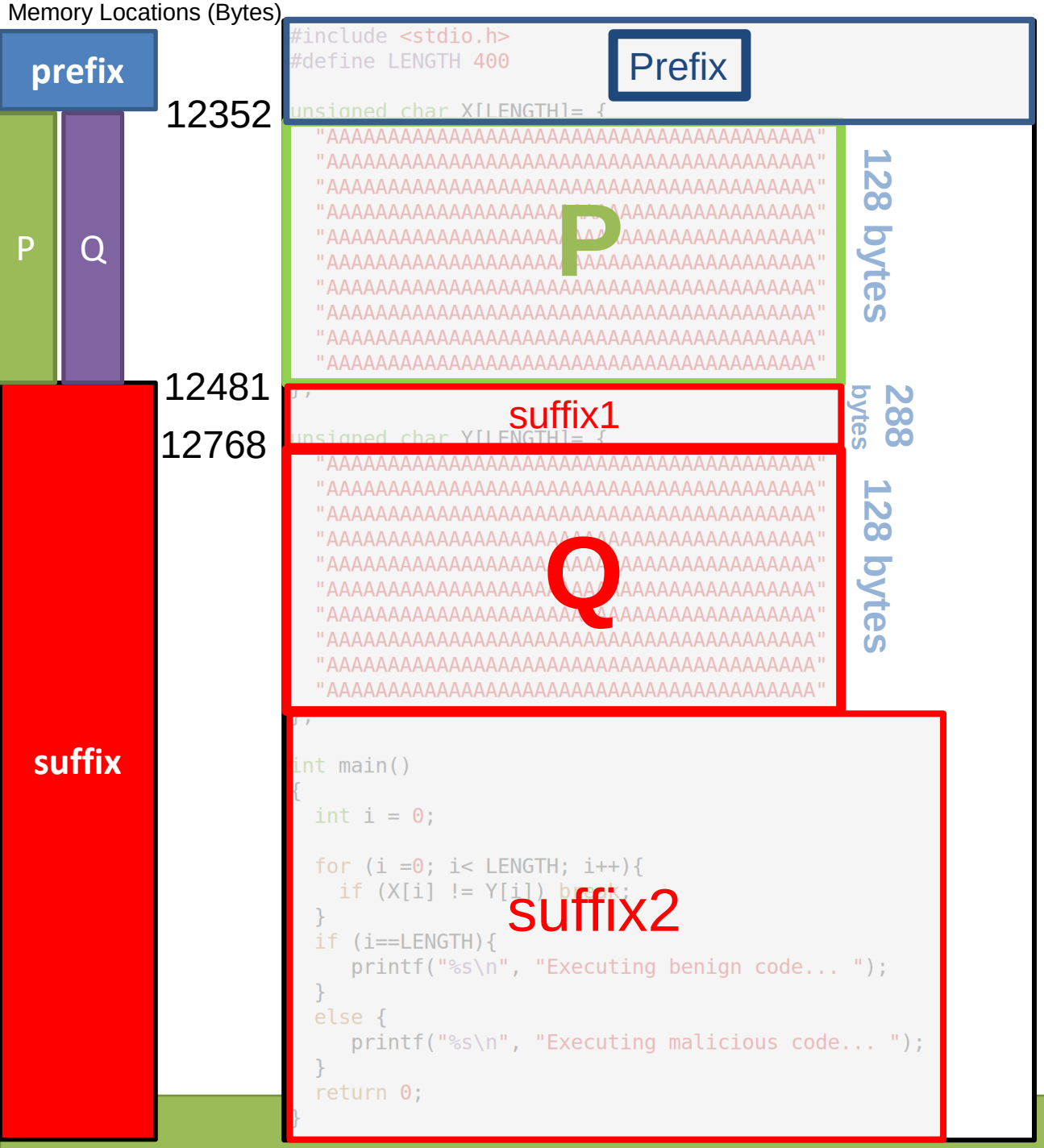
Because we just want P and Q (not the prefix), we will take the final 128 bytes of the output of `out1` and `out2`

```
[04/17/23]seed@VM:~/.../09_hashing$ tail -c 128 out1 > P
[04/17/23]seed@VM:~/.../09_hashing$ tail -c 128 out2 > Q
```



To avoid padding, we had to move the beginning of P up 32 bytes into the array X, so when we insert P/Q into array Y, we need to make sure we also do 32 bytes

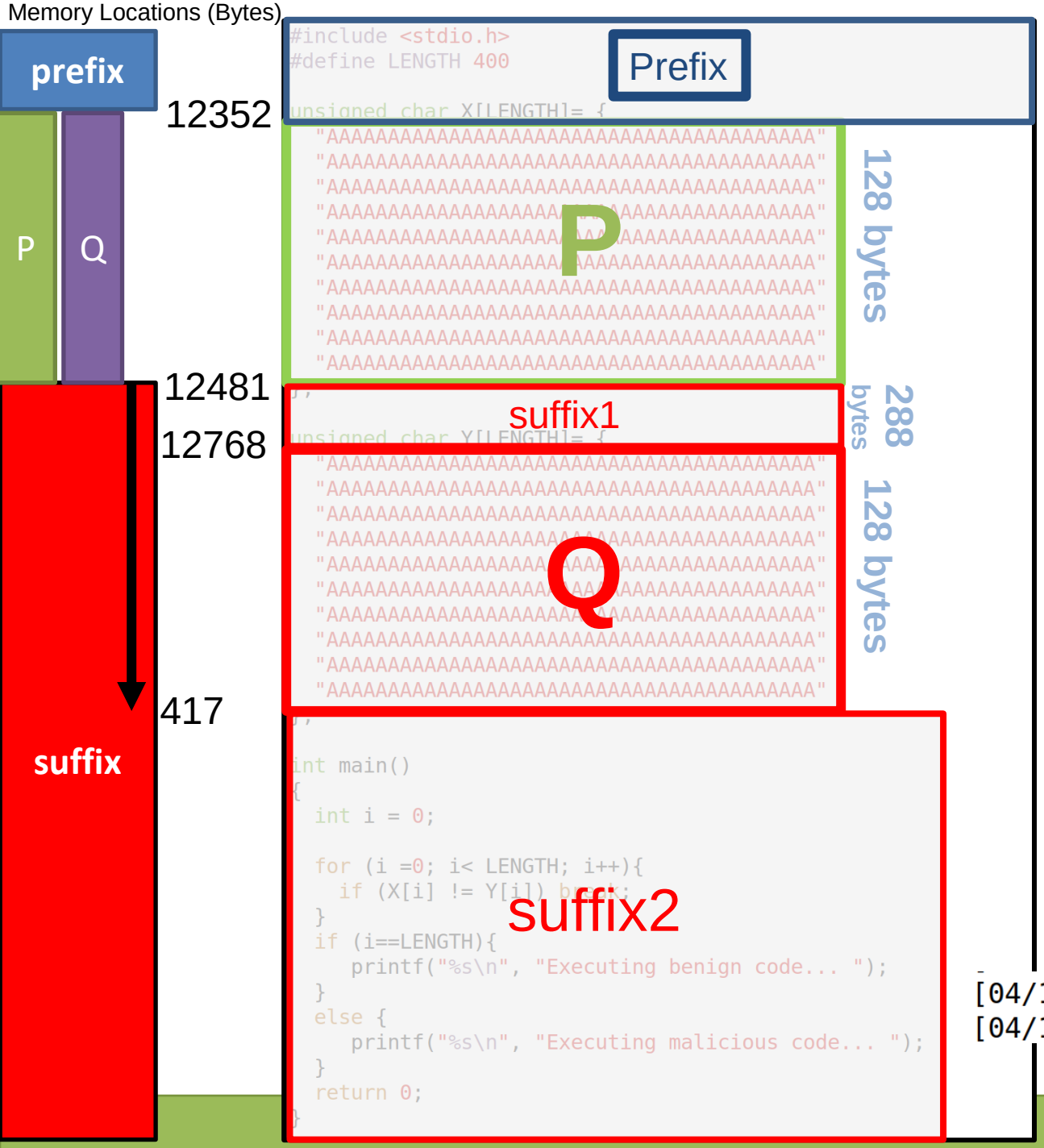
Array Y starts at 12736, but we need to inject at byte (12736 + 28) = **12768**



To avoid padding, we had to move the beginning of P up 32 bytes into the array X, so when we insert P/Q into array Y, we need to make sure we also do 32 bytes

Array Y starts at 12736, but we need to inject at byte (12736 + 28) = **12768**

Therefore, the size of suffix1 will be 12768 – 12481 = **288**



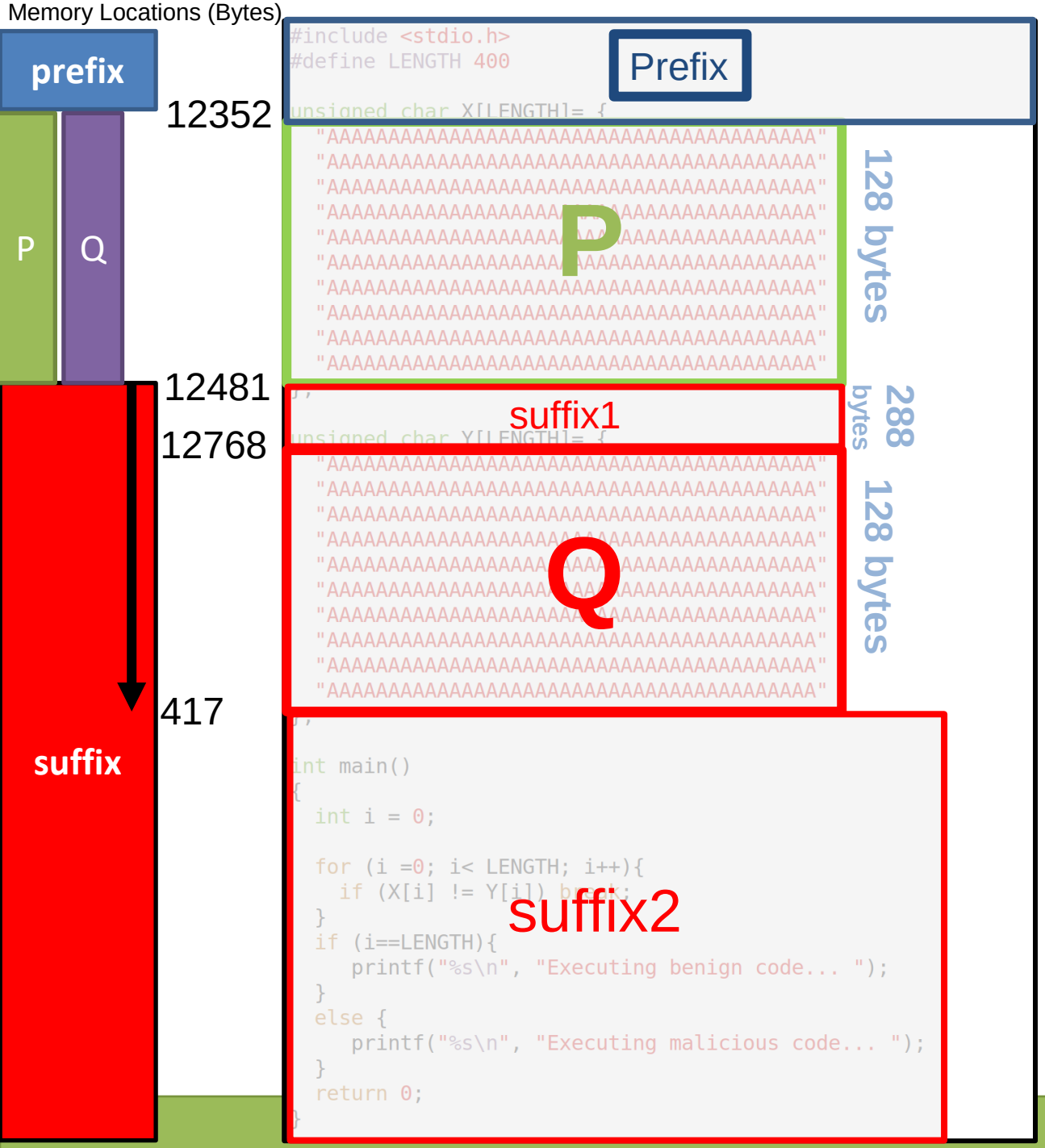
To avoid padding, we had to move the beginning of P up 32 bytes into the array X, so when we insert P/Q into array Y, we need to make sure we also do 32 bytes

Array Y starts at 12736, but we need to inject at byte (12736 + 28) = **12768**

Therefore, the size of suffix1 will be 12768 – 12481 = **288**

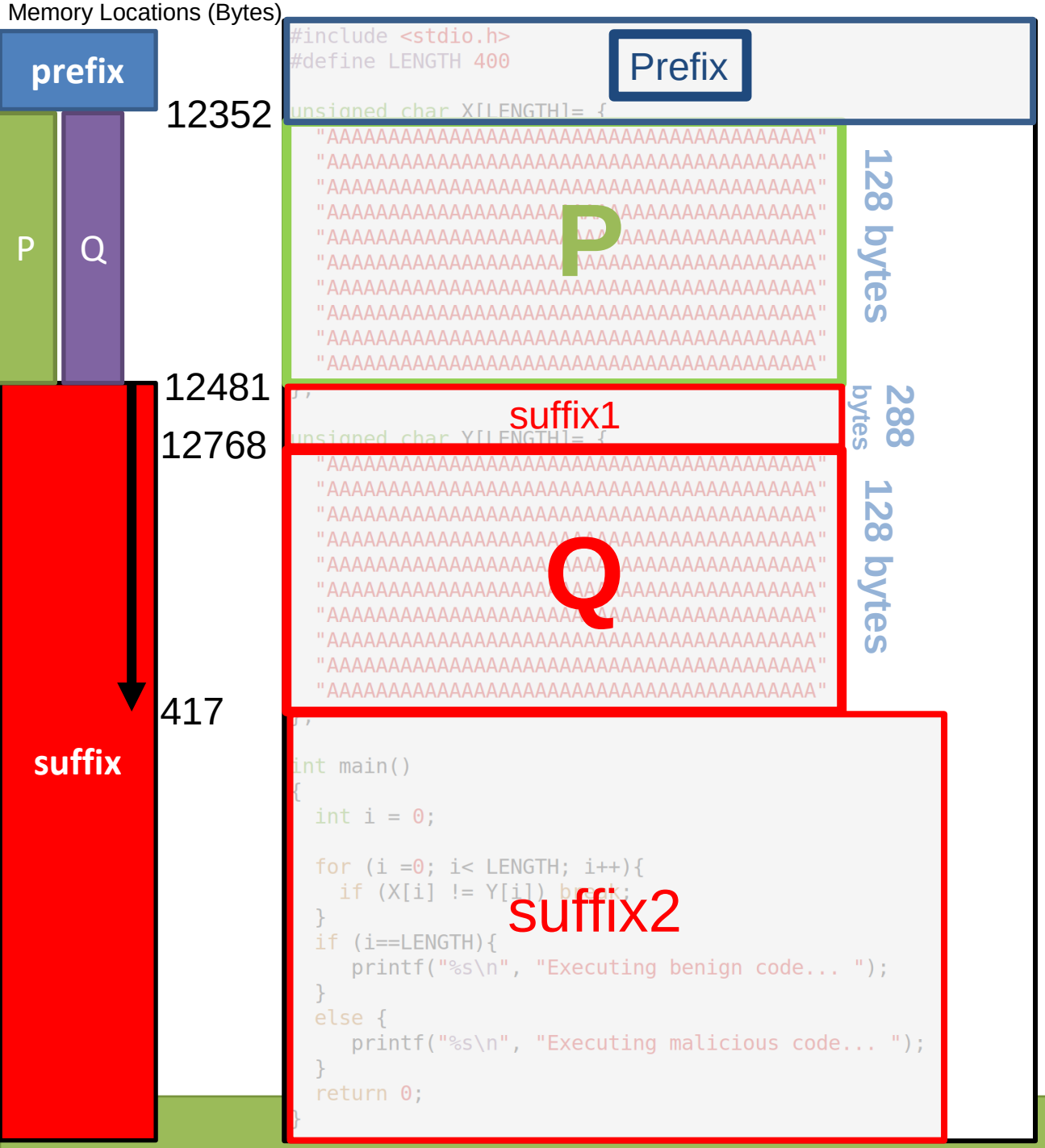
Suffix 2 will begin at byte 288 + 128 = 416 of **suffix** (but we add +1 to prevent getting an extra byte)

```
[04/17/23]seed@VM:~/.../09_hashing$ head -c 288 suffix > suffix1
[04/18/23]seed@VM:~/.../09_hashing$ head -c +417 suffx > suffix2
```



Now, we put everything together

```
[04/18/23]seed@VM:~/.../09_hashing$ cat prefix P suffix1 P suffix2 > final1
[04/18/23]seed@VM:~/.../09_hashing$ cat prefix Q suffix1 P suffix2 > final2
```

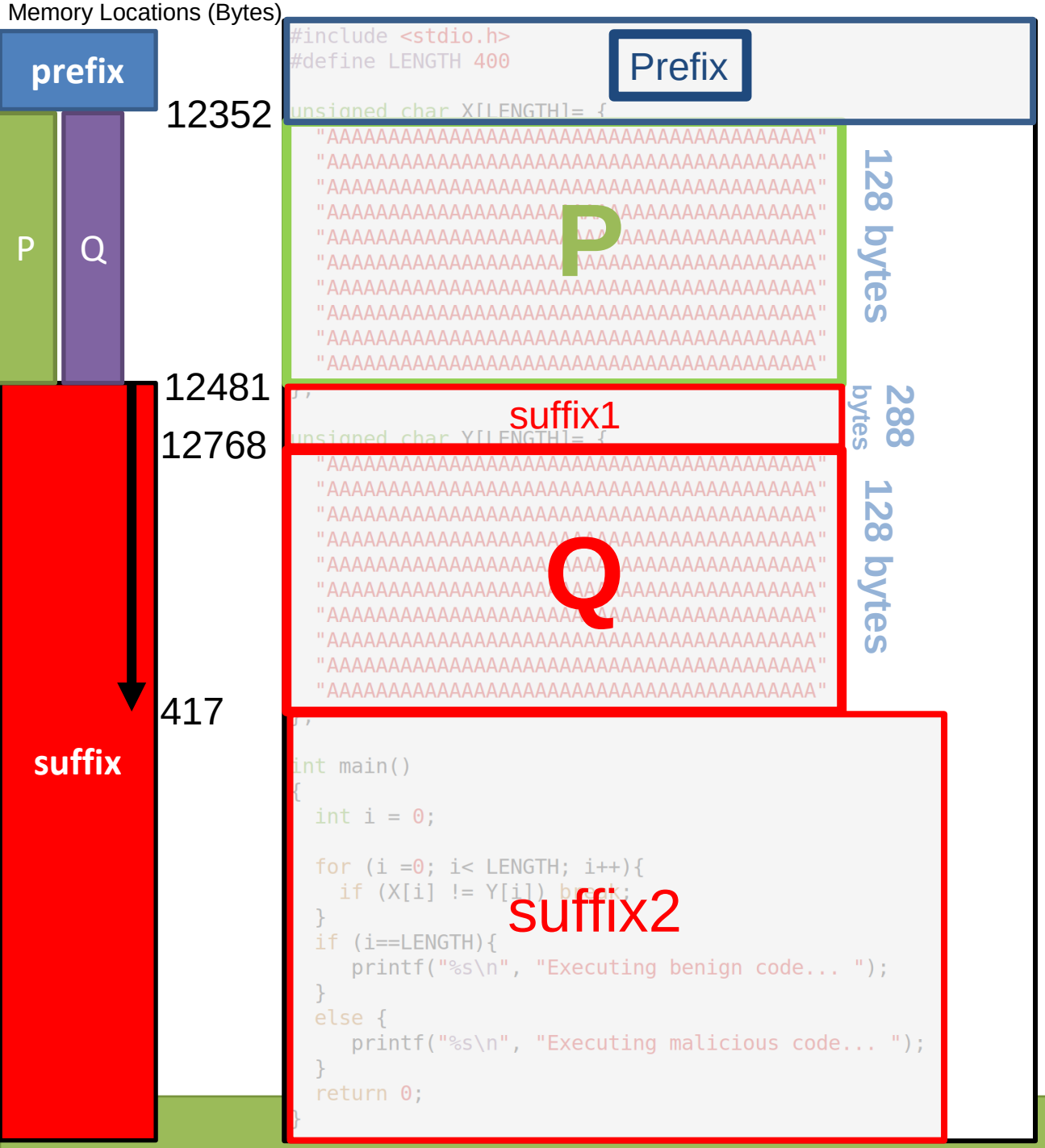


Now, we put everything together

```
[04/18/23]seed@VM:~/.../09_hashing$ cat prefix P suffix1 P suffix2 > final1
[04/18/23]seed@VM:~/.../09_hashing$ cat prefix Q suffix1 P suffix2 > final2
```

Verify that hashes match:

```
[04/18/23]seed@VM:~/.../09_hashing$ md5sum final1 final2
7eb3ea7eaefaa2efbd0ddfa0c7022e76  final1
7eb3ea7eaefaa2efbd0ddfa0c7022e76  final2
```



Now, we put everything together

```
[04/18/23]seed@VM:~/.../09_hashing$ cat prefix P suffix1 P suffix2 > final1
[04/18/23]seed@VM:~/.../09_hashing$ cat prefix Q suffix1 P suffix2 > final2
```

Verify that hashes match:

```
[04/18/23]seed@VM:~/.../09_hashing$ md5sum final1 final2
7eb3ea7eaefaa2efbd0ddfa0c7022e76  final1
7eb3ea7eaefaa2efbd0ddfa0c7022e76  final2
```

Make programs executable and run them!

```
[04/18/23]seed@VM:~/.../09_hashing$ chmod u+x final1 final2
[04/18/23]seed@VM:~/.../09_hashing$ ./final1
Executing benign code...
[04/18/23]seed@VM:~/.../09_hashing$ ./final2
Executing malicious code...
```