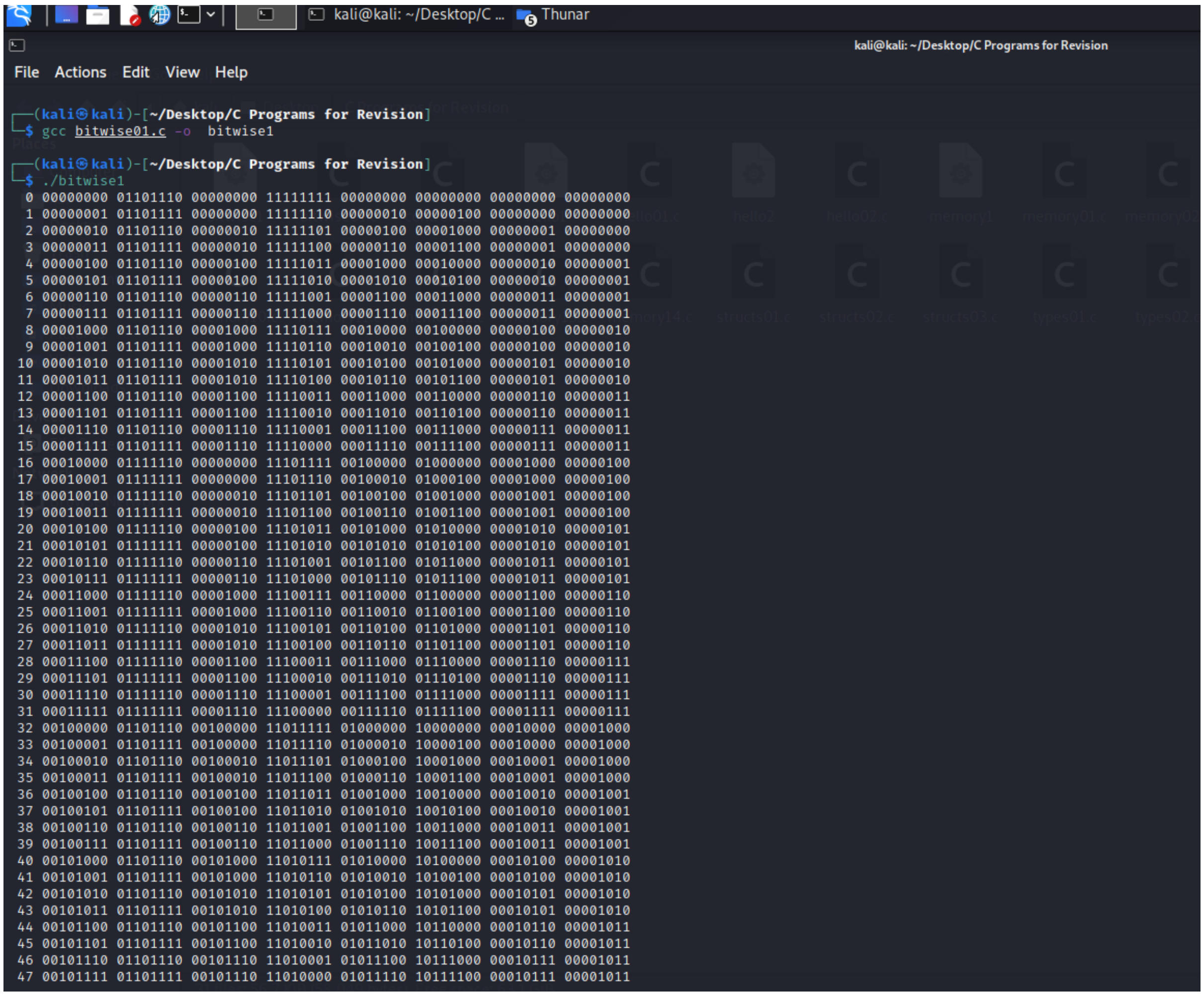
[Pre-Requisite Workshop-1]

Student Id	: NP03A190051
Student Name	: Rishab Sharma
Section	: L5CG7
Lecturer	: Jnaneshwor Bohara
Module leader	: Hiran Patel
Submitted on	: 18th September, 2021

1. bitwise01

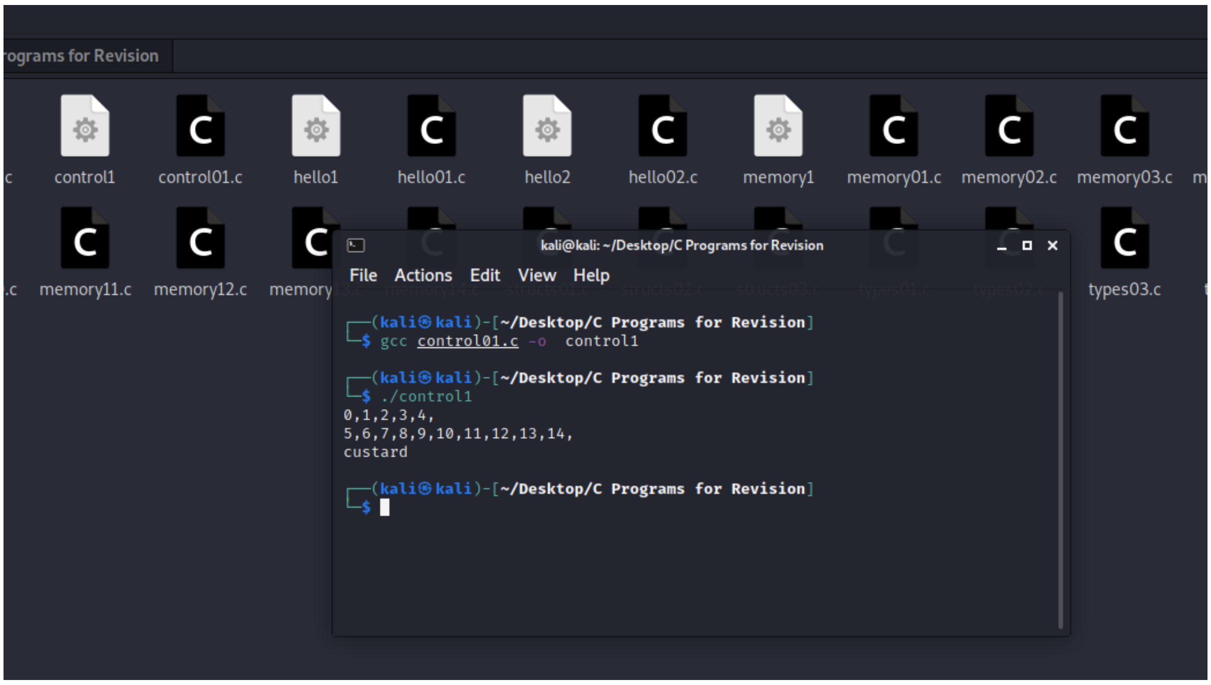
```
  #include <stdio.h>
   void toBinary(unsigned char c, unsigned char *result) {
   unsigned char i;
   unsigned char mask = 128;

  for(i=0;i<8;i++){
    if(mask & c) {
      result[i]='1';
    } else {
      result[i]='0';
    }
    mask = mask >> 1;
  }
  result[8]=0;
}
int main() {
  char result1[9];
  char result2[9];
  char result3[9];
  char result4[9];
  char result5[9];
  char result6[9];
  char result7[9];
  char result8[9];

  unsigned char n;
  unsigned char mask = 110;
  unsigned char or;
  unsigned char and;
  unsigned char not;
  unsigned char leftOnce;
  unsigned char leftTwice;
  unsigned char rightOnce;
  unsigned char rightTwice;
  int i;

  for(i=0;i<256;i++){
    n = i;
    or = n | mask;
    and = n & mask;
    not = ~n;
    leftOnce = n << 1;
    leftTwice = n << 2;
    rightOnce = n >> 1;
    rightTwice = n >> 2;
    toBinary(n, result1);
    toBinary(or, result2);
    toBinary(and, result3);
    toBinary(not, result4);
    toBinary(leftOnce, result5);
    toBinary(leftTwice, result6);
    toBinary(rightOnce, result7);
    toBinary(rightTwice, result8);
    printf("%3hu %s %s %s %s %s %s %s %s\n",
      n, result1, result2, result3, result4,
      result5, result6, result7, result8);
    n++;
  }
}
```


kali@kali: ~/Desktop/C ...
Thunar
kali@kali: ~/Desktop/C Programs for Revision
File Actions Edit View Help

```
┌──(kali㉿kali)-[~/Desktop/C Programs for Revision]
└─$ gcc bitwise01.c -o  bitwise1

┌──(kali㉿kali)-[~/Desktop/C Programs for Revision]
└─$ ./bitwise1
 0 00000000 01101110 00000000 11111111 00000000 00000000 00000000 00000000
 1 00000001 01101111 00000000 11111110 00000010 00000100 00000000 00000000
 2 00000010 01101110 00000010 11111101 00000100 00001000 00000001 00000000
 3 00000011 01101111 00000010 11111100 00000110 00001100 00000001 00000000
 4 00000100 01101110 00000100 11111011 00001000 00010000 00000010 00000001
 5 00000101 01101111 00000100 11111010 00001010 00010100 00000010 00000001
 6 00000110 01101110 00000110 11111001 00001100 00011000 00000011 00000001
 7 00000111 01101111 00000110 11111000 00001110 00011100 00000011 00000001
 8 00001000 01101110 00001000 11110111 00010000 00100000 00000100 00000010
 9 00001001 01101111 00001000 11110110 00010010 00100100 00000100 00000010
10 00001010 01101110 00001010 11110101 00010100 00101000 00000101 00000010
11 00001011 01101111 00001010 11110100 00010110 00101100 00000101 00000010
12 00001100 01101110 00001100 11110011 00011000 00110000 00000110 00000011
13 00001101 01101111 00001100 11110010 00011010 00110100 00000110 00000011
14 00001110 01101110 00001110 11110001 00011100 00111000 00000111 00000011
15 00001111 01101111 00001110 11110000 00011110 00111100 00000111 00000011
16 00010000 01111110 00000000 11101111 00100000 01000000 00001000 00000100
17 00010001 01111111 00000000 11101110 00100010 01000100 00001000 00000100
18 00010010 01111110 00000010 11101101 00100100 01001000 00001001 00000100
19 00010011 01111111 00000010 11101100 00100110 01001100 00001001 00000100
20 00010100 01111110 00000100 11101011 00101000 01010000 00001010 00000101
21 00010101 01111111 00000100 11101010 00101010 01010100 00001010 00000101
22 00010110 01111110 00000110 11101001 00101100 01011000 00001011 00000101
23 00010111 01111111 00000110 11101000 00101110 01011100 00001011 00000101
24 00011000 01111110 00001000 11100111 00110000 01100000 00001100 00000110
25 00011001 01111111 00001000 11100110 00110010 01100100 00001100 00000110
26 00011010 01111110 00001010 11100101 00110100 01101000 00001101 00000110
27 00011011 01111111 00001010 11100100 00110110 01101100 00001101 00000110
28 00011100 01111110 00001100 11100011 00111000 01110000 00001110 00000111
29 00011101 01111111 00001100 11100010 00111010 01110100 00001110 00000111
30 00011110 01111110 00001110 11100001 00111100 01111000 00001111 00000111
31 00011111 01111111 00001110 11100000 00111110 01111100 00001111 00000111
32 00100000 01101110 00100000 11011111 01000000 10000000 00010000 00001000
33 00100001 01101111 00100000 11011110 01000010 10000100 00010000 00001000
34 00100010 01101110 00100010 11011101 01000100 10001000 00010001 00001000
35 00100011 01101111 00100010 11011100 01000110 10001100 00010001 00001000
36 00100100 01101110 00100100 11011011 01001000 10010000 00010010 00001001
37 00100101 01101111 00100100 11011010 01001010 10010100 00010010 00001001
38 00100110 01101110 00100110 11011001 01001100 10011000 00010011 00001001
39 00100111 01101111 00100110 11011000 01001110 10011100 00010011 00001001
40 00101000 01101110 00101000 11010111 01010000 10100000 00010100 00001010
41 00101001 01101111 00101000 11010110 01010010 10100100 00010100 00001010
42 00101010 01101110 00101010 11010101 01010100 10101000 00010101 00001010
43 00101011 01101111 00101010 11010100 01010110 10101100 00010101 00001010
44 00101100 01101110 00101100 11010011 01011000 10110000 00010110 00001011
45 00101101 01101111 00101100 11010010 01011010 10110100 00010110 00001011
46 00101110 01101110 00101110 11010001 01011100 10111000 00010111 00001011
47 00101111 01101111 00101110 11010000 01011110 10111100 00010111 00001011
```

2. control01

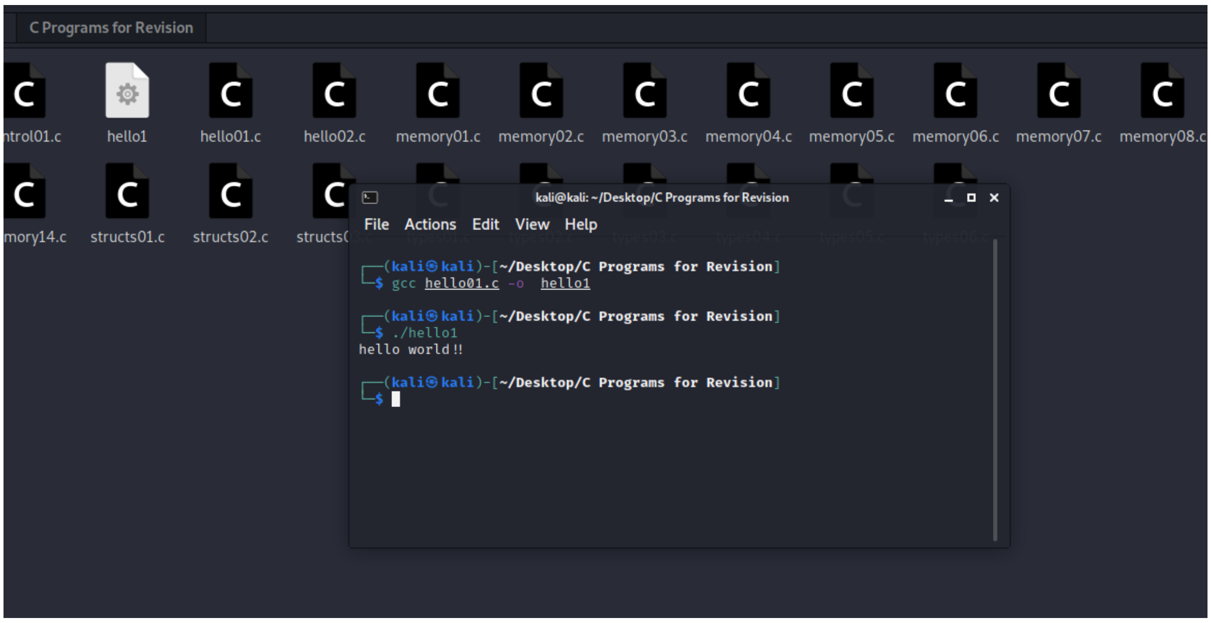
```
#include <stdlib.h>
#include <stdio.h>

int main() {
  int i;
  for(i=0;i<5;i++){
    printf("%d,", i);
  }
  printf("\n");

  while(i<10){
    printf("%d,", i);
    i++;
  }
  do {
    printf("%d,", i);
    i++;
  } while(i<15);
  printf("\n");

  if(i>13){
    printf("custard\n");
  } else {
    printf("gravy\n");
  }

  return EXIT_SUCCESS;
}
```



```
┌──(kali㉿kali)-[~/Desktop/C Programs for Revision]
└─$ gcc control01.c -o  control1

┌──(kali㉿kali)-[~/Desktop/C Programs for Revision]
└─$ ./control1
0,1,2,3,4,
5,6,7,8,9,10,11,12,13,14,
custard

┌──(kali㉿kali)-[~/Desktop/C Programs for Revision]
└─$ 
```

3. hello01

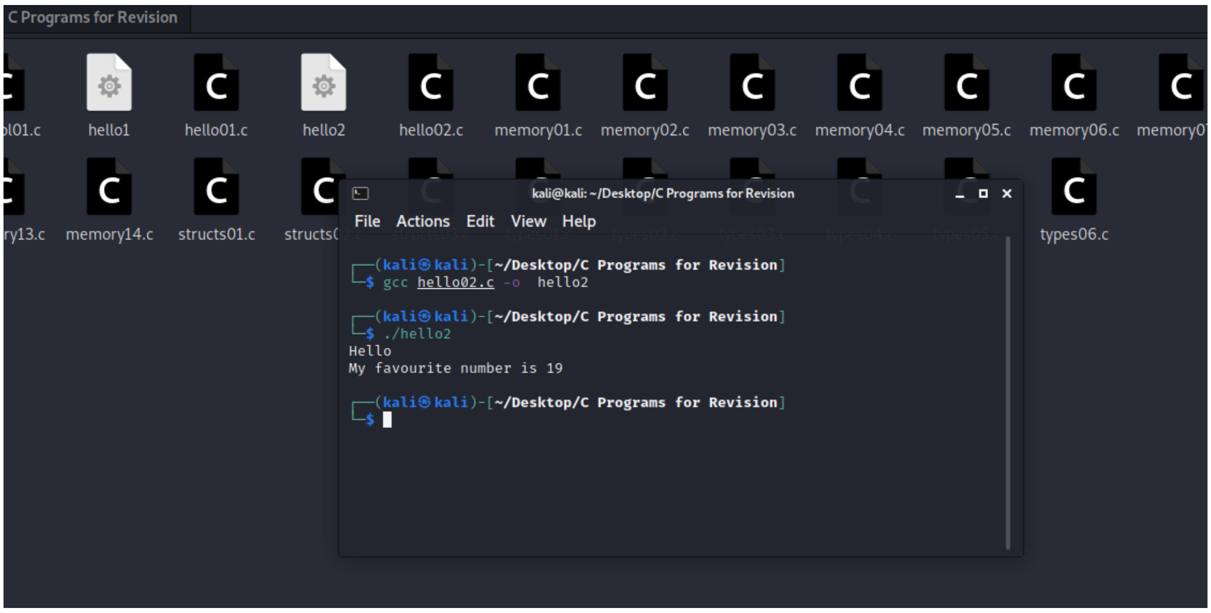
```
#include <stdio.h>

int main(){
  printf("hello world!!\n");
  return 0;
}
```



```
┌──(kali㉿kali)-[~/Desktop/C Programs for Revision]
└─$ gcc hello01.c -o  hello1

┌──(kali㉿kali)-[~/Desktop/C Programs for Revision]
└─$ ./hello1
hello world !!

┌──(kali㉿kali)-[~/Desktop/C Programs for Revision]
└─$ 
```

4. hello02

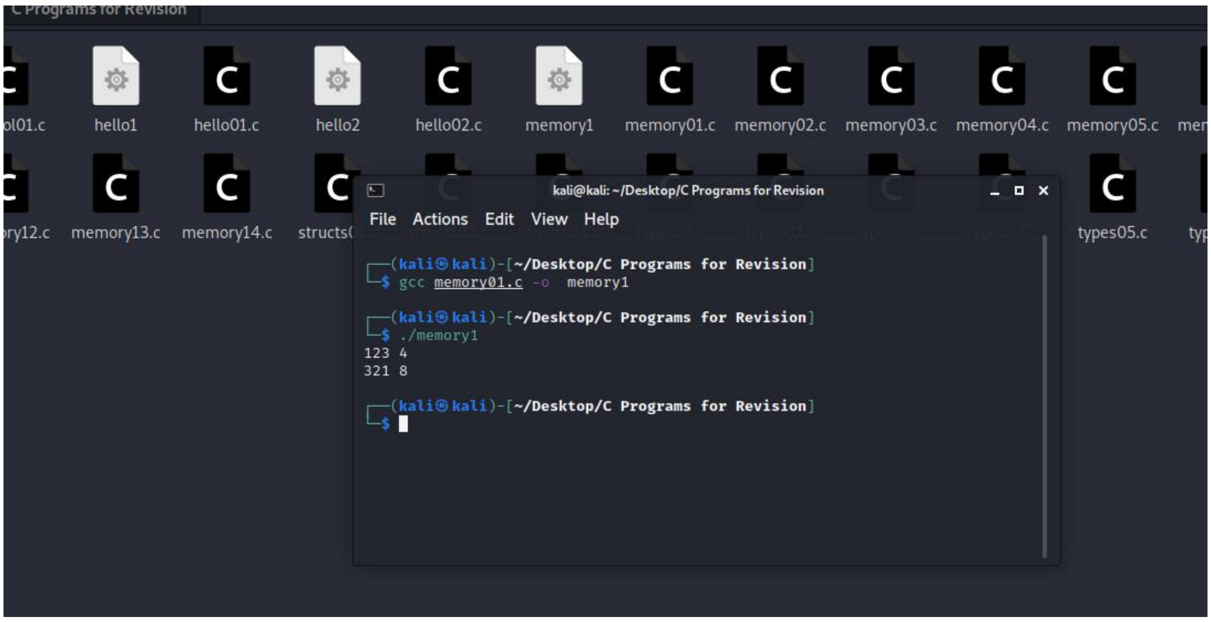
```
#include <stdlib.h>
#include <stdio.h>

int main() {
  int n = 19;
  printf("Hello\nMy favourite number is %d\n", n);
  return EXIT_SUCCESS;
}
```



```
┌──(kali㉿kali)-[~/Desktop/C Programs for Revision]
└─$ gcc hello02.c -o  hello2

┌──(kali㉿kali)-[~/Desktop/C Programs for Revision]
└─$ ./hello2
Hello
My favourite number is 19

┌──(kali㉿kali)-[~/Desktop/C Programs for Revision]
└─$ 
```

5. memory01

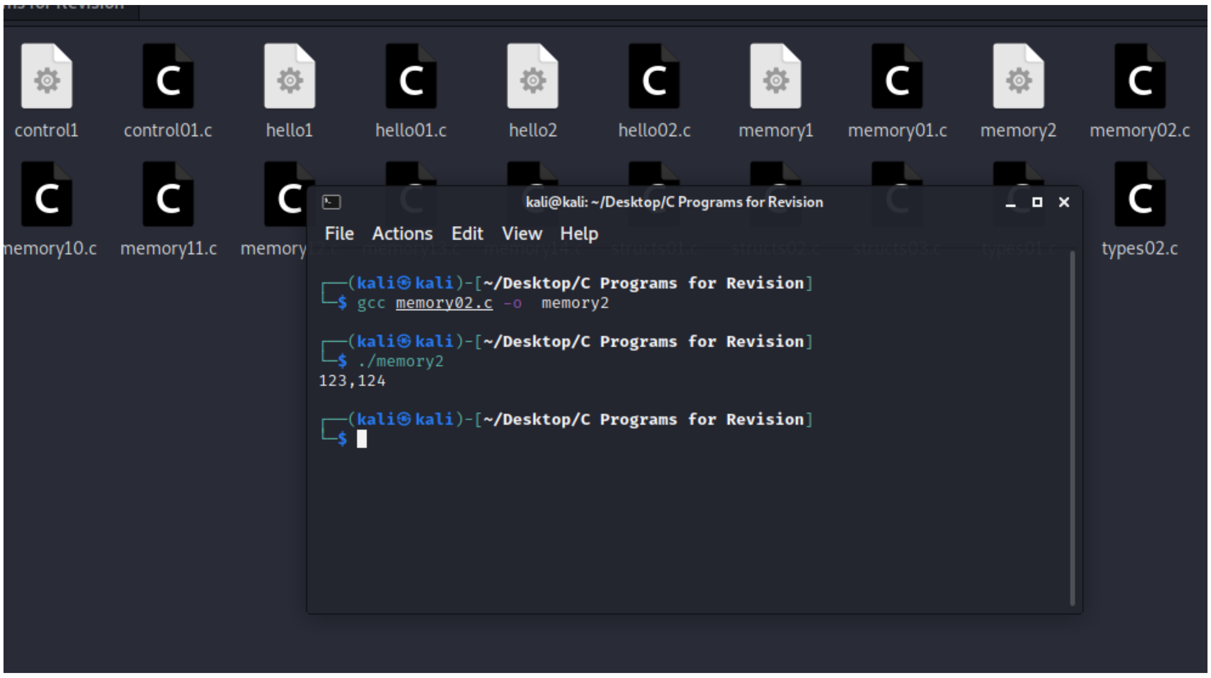
```
#include <stdio.h>
#include <stdlib.h>

int main() {
  int x = 123;
  long int y = 321;
  printf("%d %ld\n",  x, sizeof(x));
  printf("%ld %ld\n", y, sizeof(y));
  return EXIT_SUCCESS;
}
```



```
┌──(kali㉿kali)-[~/Desktop/C Programs for Revision]
└─$ gcc memory01.c -o  memory1

┌──(kali㉿kali)-[~/Desktop/C Programs for Revision]
└─$ ./memory1
123 4
321 8

┌──(kali㉿kali)-[~/Desktop/C Programs for Revision]
└─$ 
```

6. memory02

```
#include <stdio.h>
#include <stdlib.h>

int inc(int w) {
  return w + 1;
}
int main() {
    int x = 123;
    int y = inc(x);
    printf("%d,%d\n", x, y);
    return EXIT_SUCCESS;
  }
```

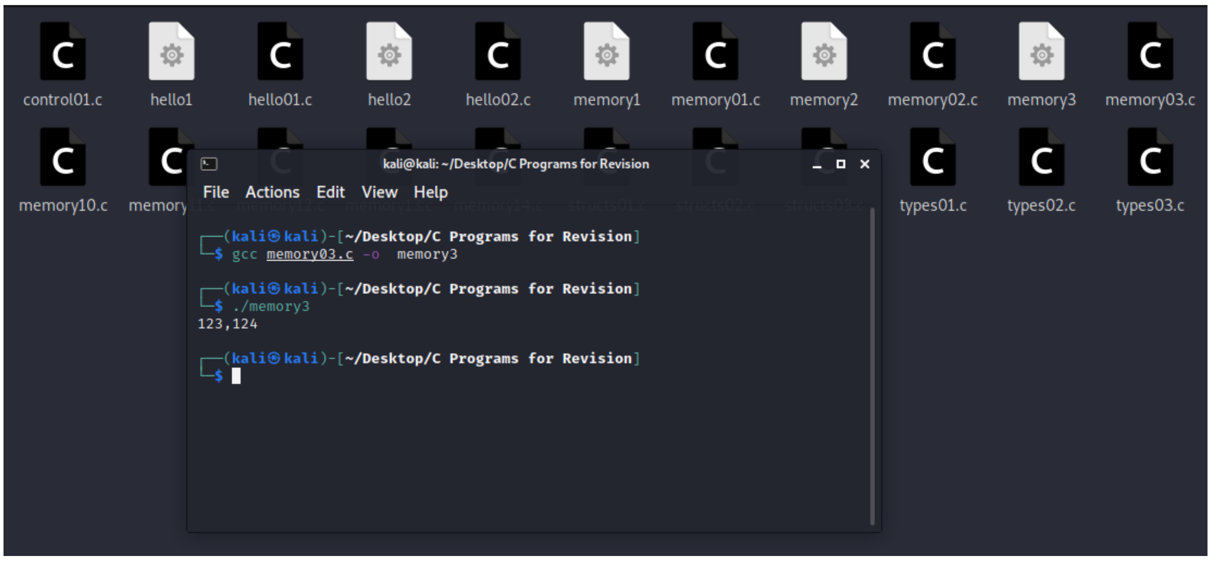


```
┌──(kali㉿kali)-[~/Desktop/C Programs for Revision]
└─$ gcc memory02.c -o  memory2

┌──(kali㉿kali)-[~/Desktop/C Programs for Revision]
└─$ ./memory2
123,124

┌──(kali㉿kali)-[~/Desktop/C Programs for Revision]
└─$ 
```

7. memory03

```
#include <stdio.h>
#include <stdlib.h>

void inc(int *w) {
  *w = *w + 1;
}
int main() {
  int x = 123;
  int y = x;
  inc(&y);
  printf("%d,%d\n", x, y);
  return EXIT_SUCCESS;
```

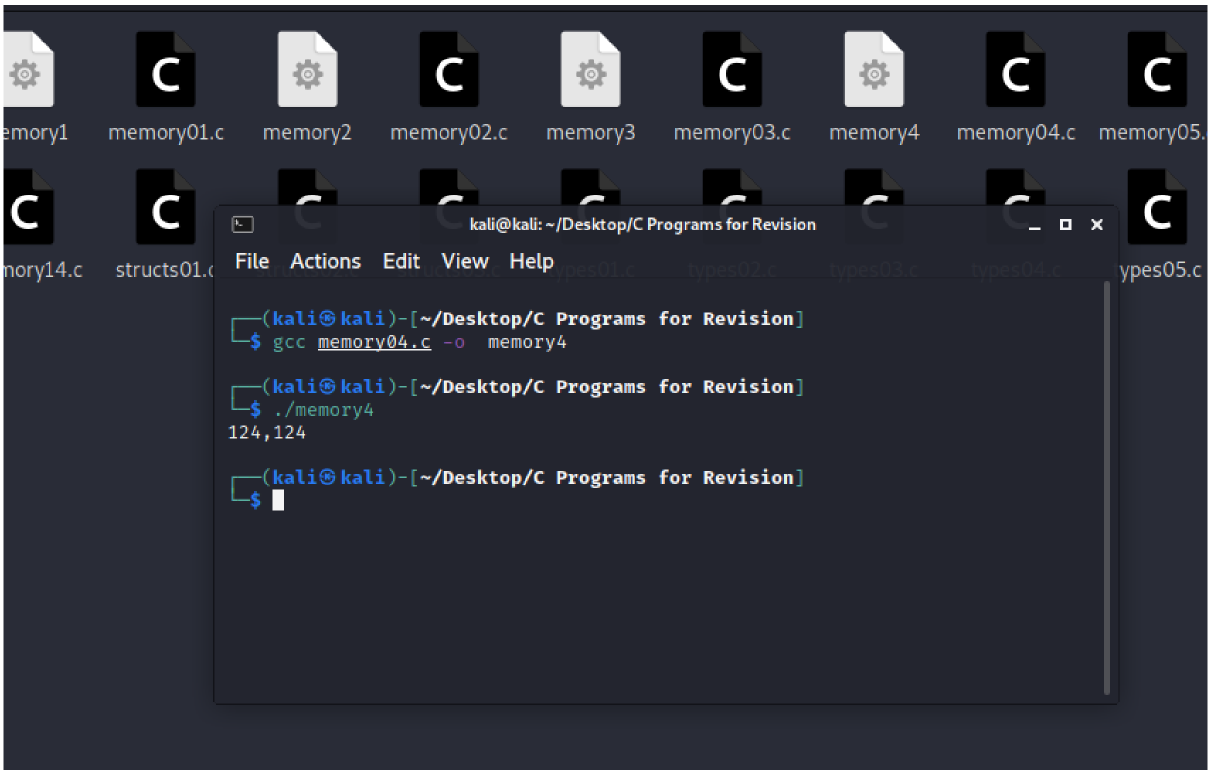
}



```
┌──(kali㉿kali)-[~/Desktop/C Programs for Revision]
└─$ gcc memory03.c -o memory3

┌──(kali㉿kali)-[~/Desktop/C Programs for Revision]
└─$ ./memory3
123,124

┌──(kali㉿kali)-[~/Desktop/C Programs for Revision]
└─$ 
```

8. memory04

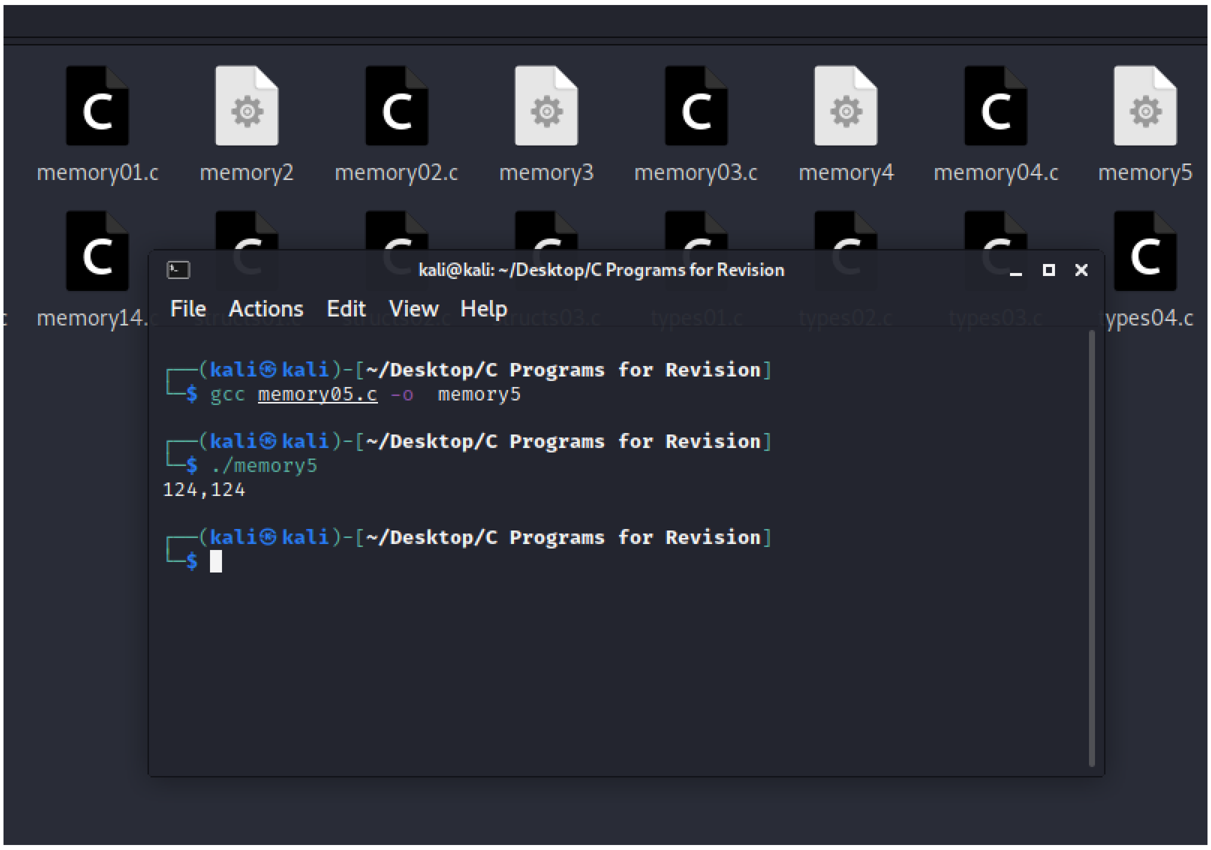
```
#include <stdio.h>
#include <stdlib.h>

void inc(int *w) {
  *w = *w + 1;
}

int main() {
  int x = 123;
  int *y;
  y = &x;
  inc(y);
  printf("%d,%d\n", x, *y);
  return EXIT_SUCCESS;
}
```



```
┌──(kali㉿kali)-[~/Desktop/C Programs for Revision]
└─$ gcc memory04.c -o  memory4

┌──(kali㉿kali)-[~/Desktop/C Programs for Revision]
└─$ ./memory4
124,124

┌──(kali㉿kali)-[~/Desktop/C Programs for Revision]
└─$ 
```

9. memory05

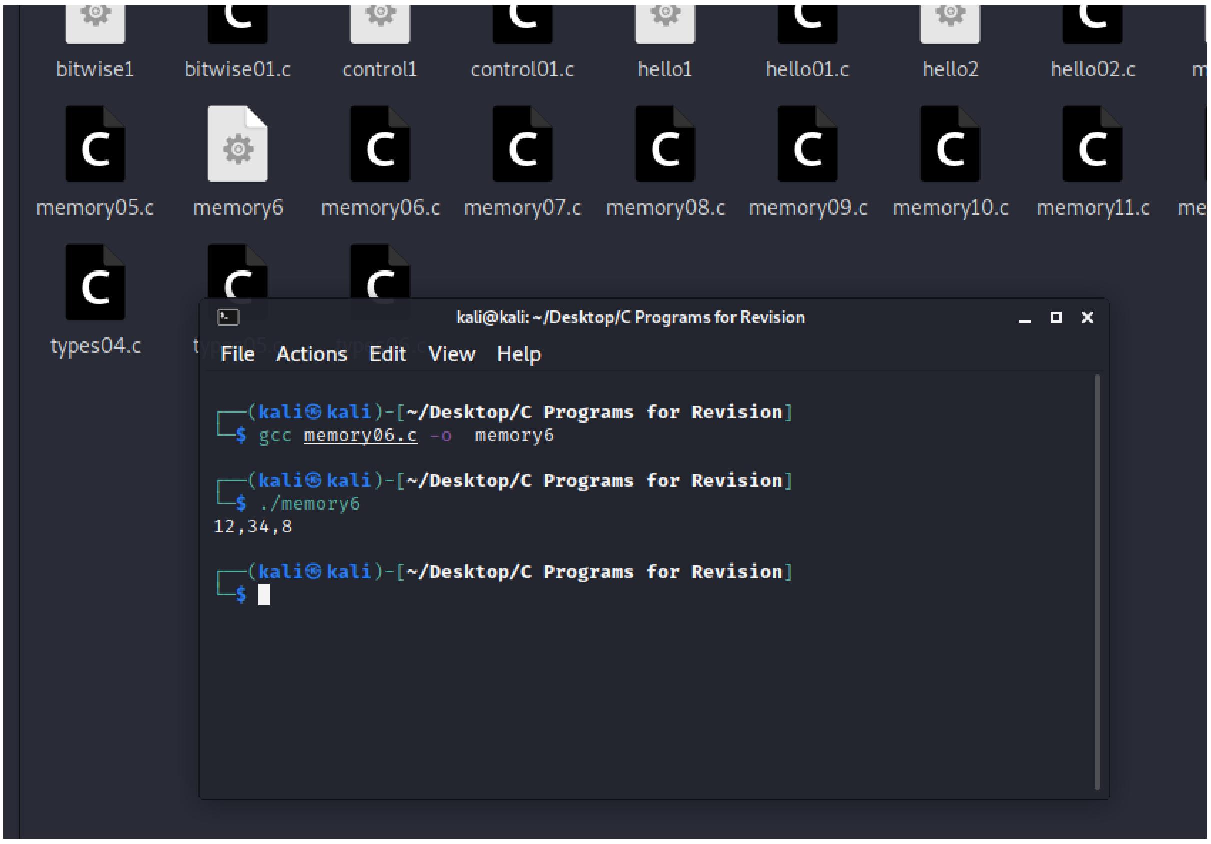
```
#include <stdio.h>
#include <stdlib.h>
#include <malloc.h>
void inc(int *w) {
  *w = *w + 1;
}
int main() {
  int *x = malloc(sizeof(int));
  *x = 123;
  int *y;
  y = x;
  inc(y);
  printf("%d,%d\n", *x, *y);
  free(x);
  return EXIT_SUCCESS;
}
```



```
┌──(kali㉿kali)-[~/Desktop/C Programs for Revision]
└─$ gcc memory05.c -o  memory5

┌──(kali㉿kali)-[~/Desktop/C Programs for Revision]
└─$ ./memory5
124,124

┌──(kali㉿kali)-[~/Desktop/C Programs for Revision]
└─$ 
```

10. memory06

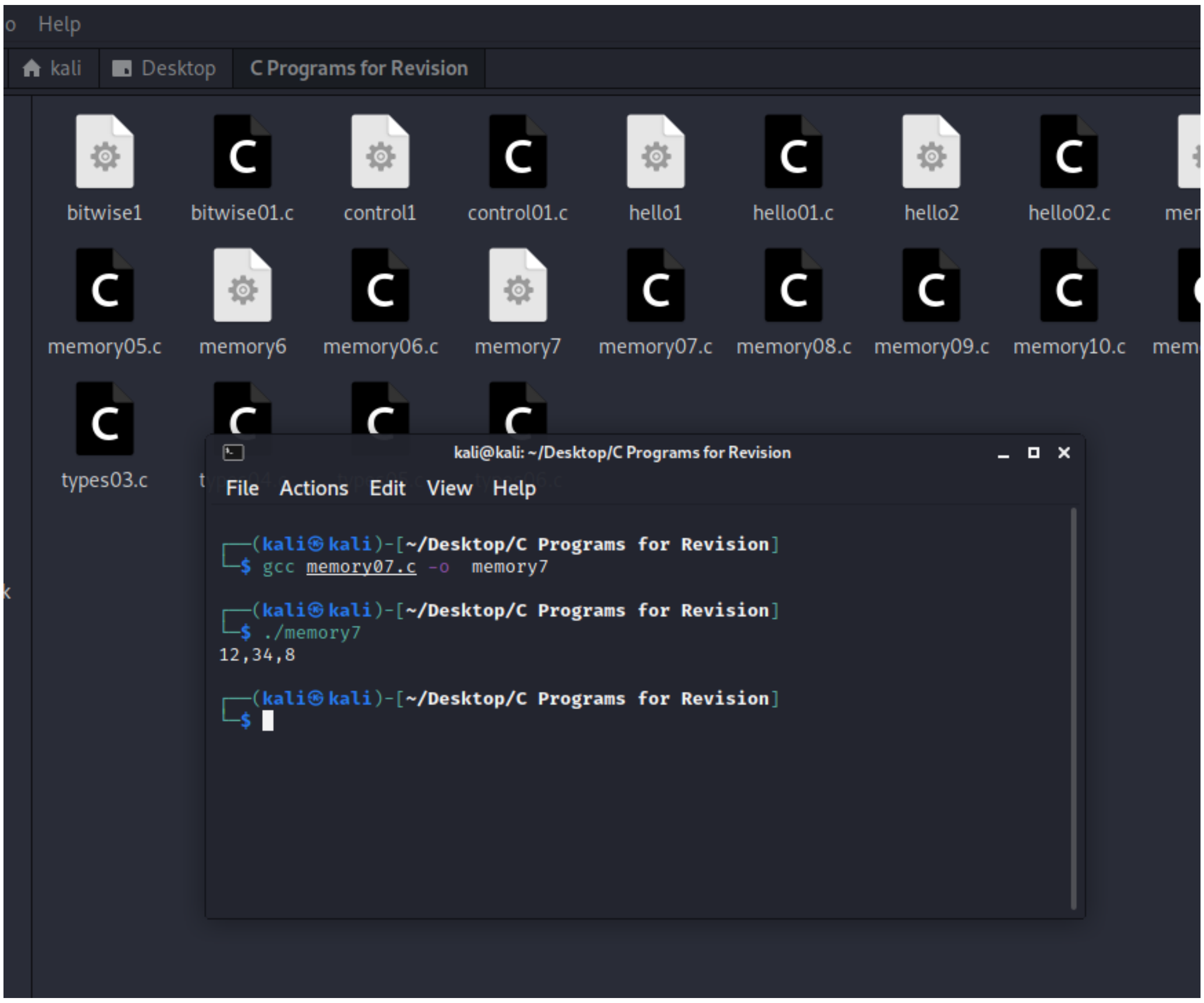
```
#include <stdio.h>
#include <stdlib.h>

struct pair {
  int a;
  int b;
};

int main() {
  struct pair x;
  x.a = 12;
  x.b = 34;
  printf("%d,%d,%ld\n", x.a, x.b, sizeof(struct pair));
  return EXIT_SUCCESS;
}
```



```
┌──(kali㉿kali)-[~/Desktop/C Programs for Revision]
└─$ gcc memory06.c -o  memory6

┌──(kali㉿kali)-[~/Desktop/C Programs for Revision]
└─$ ./memory6
12,34,8

┌──(kali㉿kali)-[~/Desktop/C Programs for Revision]
└─$ 
```

11. memory07

```
#include <stdio.h>
#include <stdlib.h>
typedef struct  {
  int a;
  int b;
} pair;

int main() {
  pair x;
  x.a = 12;
  x.b = 34;
  printf("%d,%d,%ld\n", x.a, x.b, sizeof(pair));
  return EXIT_SUCCESS;
}
```

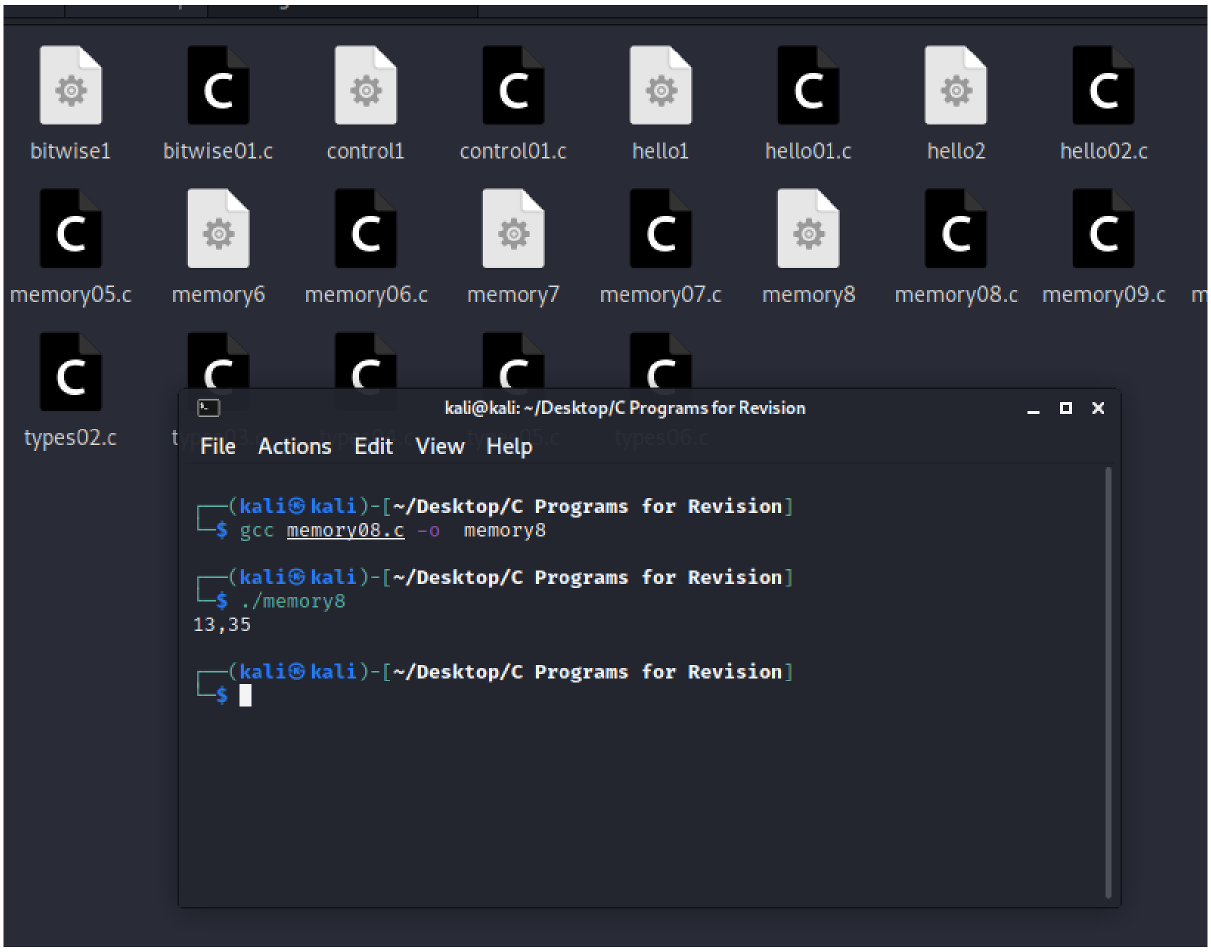


```
┌──(kali㉿kali)-[~/Desktop/C Programs for Revision]
└─$ gcc memory07.c -o  memory7

┌──(kali㉿kali)-[~/Desktop/C Programs for Revision]
└─$ ./memory7
12,34,8

┌──(kali㉿kali)-[~/Desktop/C Programs for Revision]
└─$ 
```

12. memory08

```
#include <stdio.h>
#include <stdlib.h>

typedef struct {
  int a;
  int b;
} pair;

void inc(pair *w) {
  w->a = w->a + 1;
  w->b = w->b + 1;
}
```

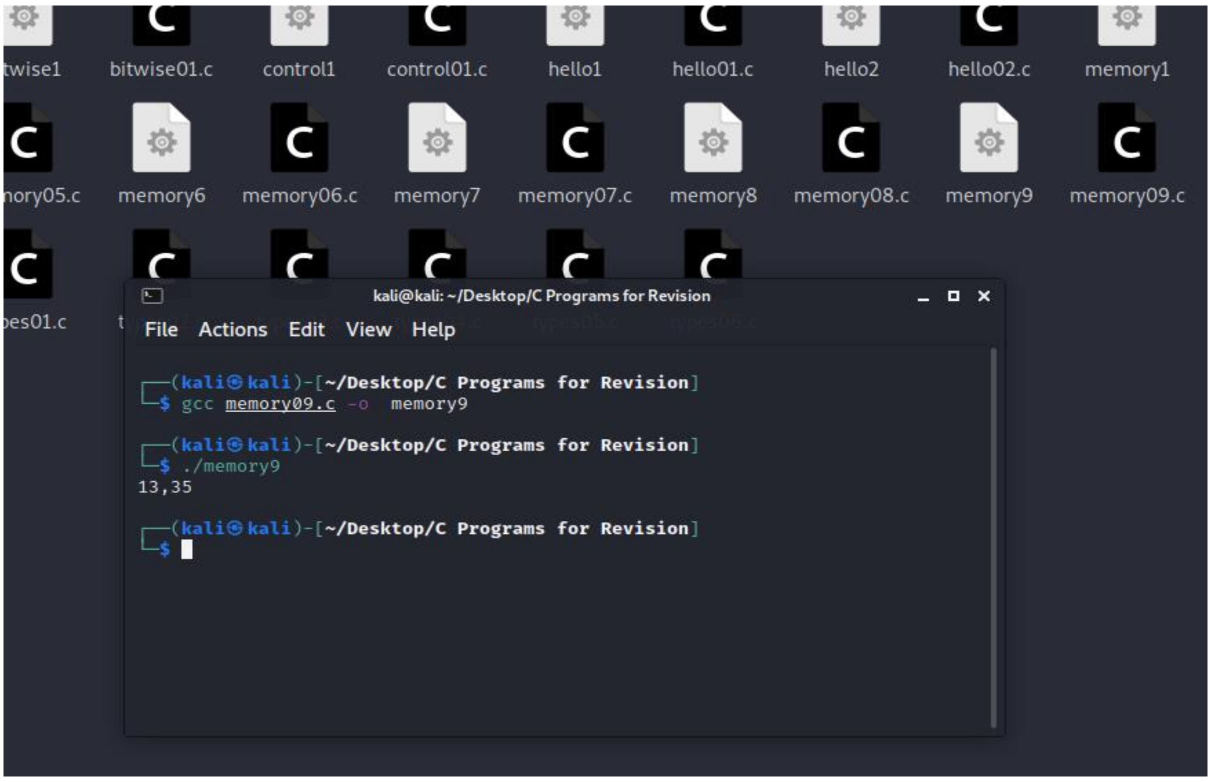
```
int main() {
  pair x;
  x.a = 12;
  x.b = 34;
  inc(&x);
  printf("%d,%d\n", x.a, x.b);
  return EXIT_SUCCESS;
}
```



```
┌──(kali㉿kali)-[~/Desktop/C Programs for Revision]
└─$ gcc memory08.c -o  memory8

┌──(kali㉿kali)-[~/Desktop/C Programs for Revision]
└─$ ./memory8
13,35

┌──(kali㉿kali)-[~/Desktop/C Programs for Revision]
└─$ 
```

13. memory09

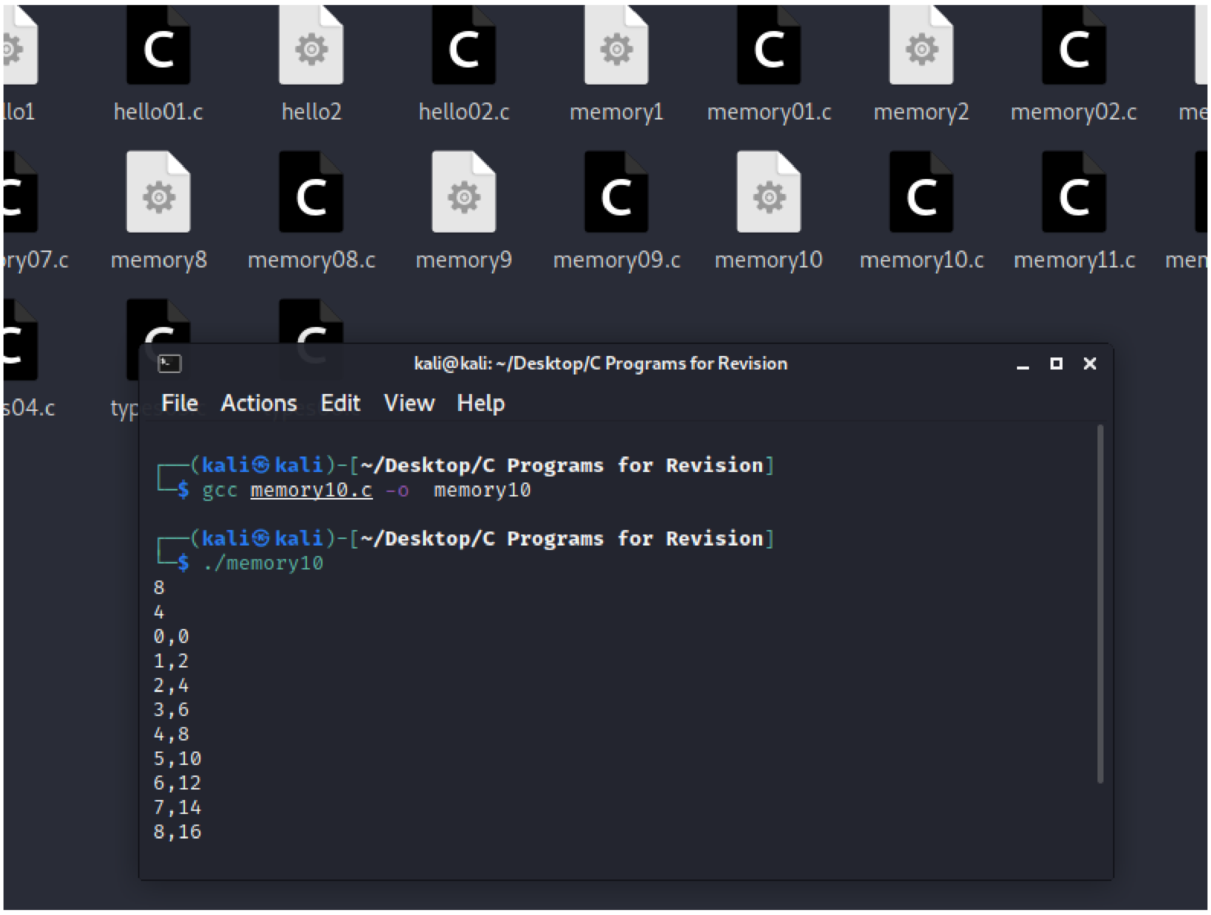
```
#include <stdio.h>
#include <stdlib.h>
#include <malloc.h>

typedef struct {
  int a;
  int b;
} pair;

void inc(pair *w) {
  w->a = w->a + 1;
  w->b = w->b + 1;
}
int main() {
  pair *x;
  x = malloc(sizeof(pair));
  x->a = 12;
  x->b = 34;
  inc(x);
  printf("%d,%d\n", x->a, x->b);
  free(x);
  return EXIT_SUCCESS;
}
```



```
┌──(kali㉿kali)-[~/Desktop/C Programs for Revision]
└─$ gcc memory09.c -o  memory9

┌──(kali㉿kali)-[~/Desktop/C Programs for Revision]
└─$ ./memory9
13,35

┌──(kali㉿kali)-[~/Desktop/C Programs for Revision]
└─$ 
```

14. memory10

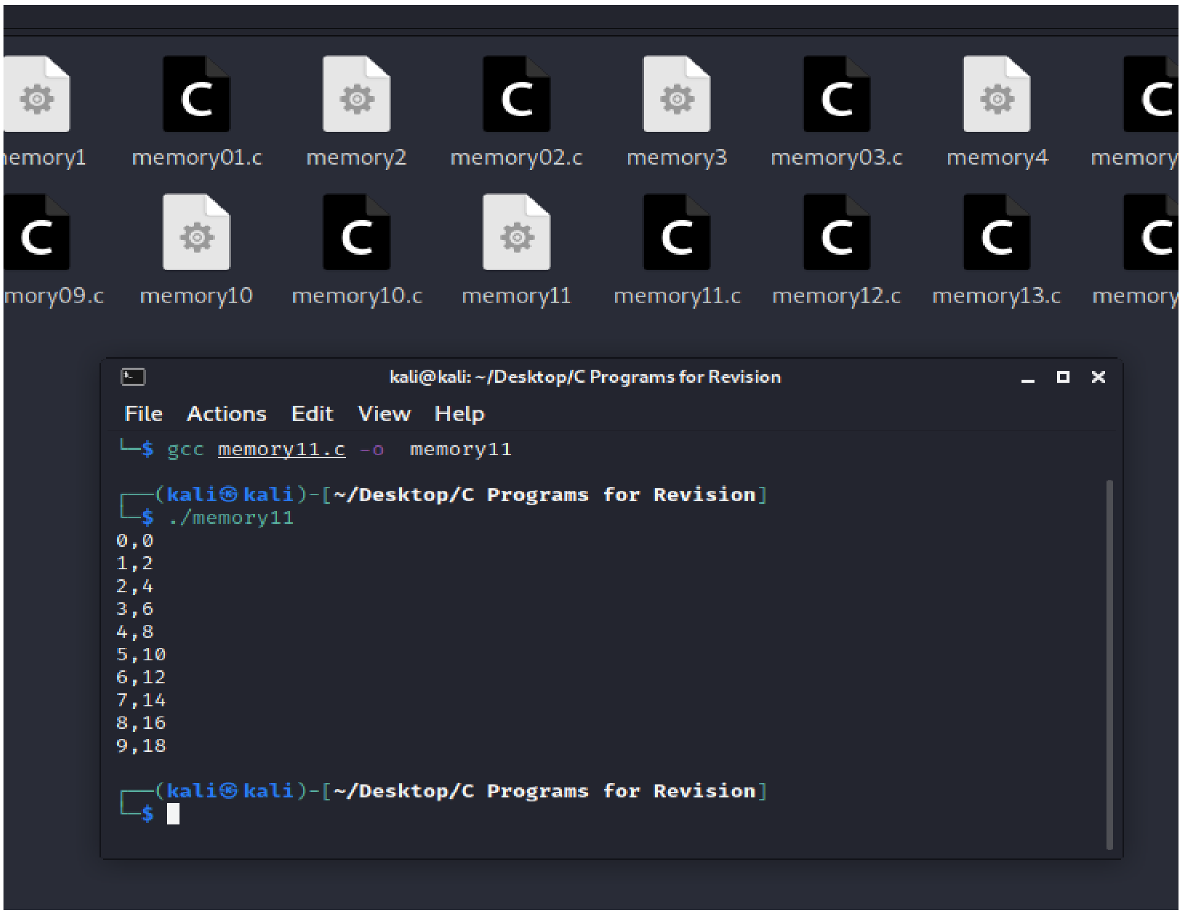
```
#include <stdio.h>
#include <stdlib.h>
#include <malloc.h>

int n = 10;

int main() {
  int i;
  int *x;
  x = malloc(sizeof(int) * n);

  printf("%ld\n", sizeof(x));
  printf("%ld\n", sizeof(*x));

  for(i=0;i<n;i++){
    x[i] = 2 * i;
  }

  for(i=0;i<n;i++){
    printf("%d,%d\n", i, x[i]);
  }

  free(x);
  return EXIT_SUCCESS;
}
```



```
┌──(kali㉿kali)-[~/Desktop/C Programs for Revision]
└─$ gcc memory10.c -o  memory10

┌──(kali㉿kali)-[~/Desktop/C Programs for Revision]
└─$ ./memory10
8
4
0,0
1,2
2,4
3,6
4,8
5,10
6,12
7,14
8,16
```

15. memory11

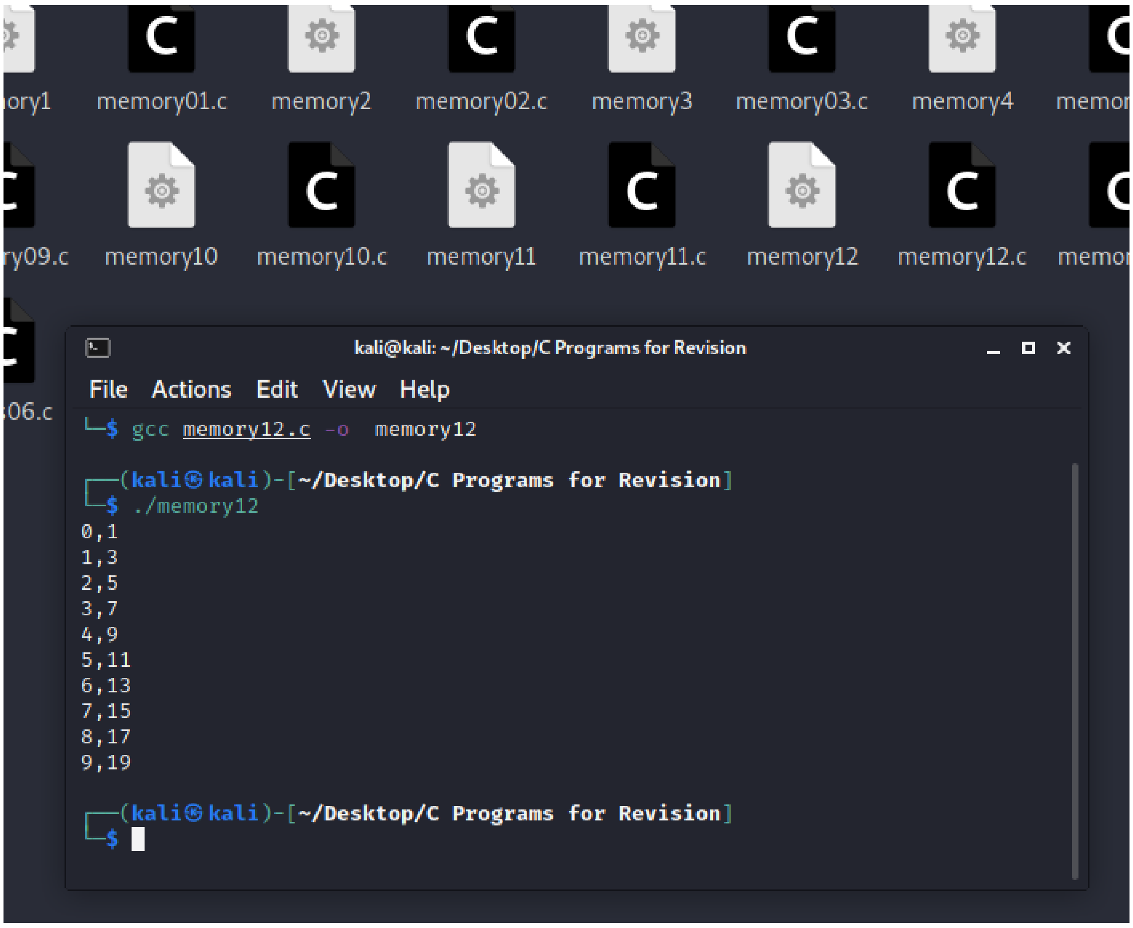
```
#include <stdio.h>
#include <stdlib.h>
#include <malloc.h>
int n = 10;

int main() {
 int i;
 int *x, *y;
 x = malloc(sizeof(int) * n);
 y = x;

 for(i=0;i<n;i++){
  *y = 2 * i;
  y++;
 }

 y = x;

 for(i=0;i<n;i++){
  printf("%d,%d\n", i, *y);
  y++;
 }

 free(x);
 return EXIT_SUCCESS;
}
```



```
└─$ gcc memory11.c -o  memory11

┌──(kali㉿kali)-[~/Desktop/C Programs for Revision]
└─$ ./memory11
0,0
1,2
2,4
3,6
4,8
5,10
6,12
7,14
8,16
9,18

┌──(kali㉿kali)-[~/Desktop/C Programs for Revision]
└─$ 
```

16. memory12

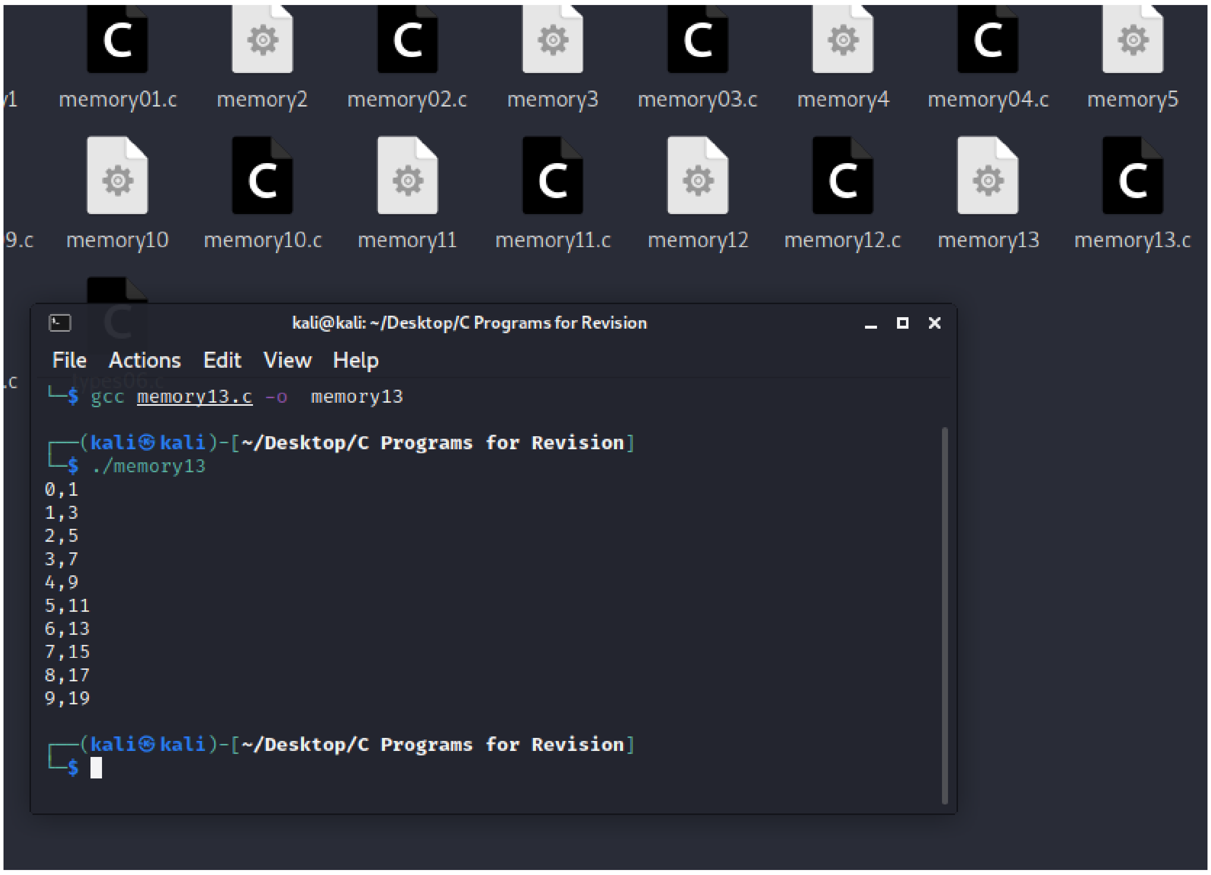
```
#include <stdio.h>
#include <stdlib.h>
#include <malloc.h>

int n = 10;

void inc(int *w) {
  int i;
  for(i=0;i<n;i++){
    w[i] = w[i] + 1;
  }
}

int main() {
  int i;
  int *x, *y;
  x = malloc(sizeof(int) * n);
  y = x;

  for(i=0;i<n;i++){
    *y = 2 * i;
    y++;
  }

  inc(x);
  y = x;

  for(i=0;i<n;i++){
    printf("%d,%d\n", i, *y);
    y++;
  }
  free(x);
  return EXIT_SUCCESS;
}
```



```
└─$ gcc memory12.c -o  memory12

┌──(kali㉿kali)-[~/Desktop/C Programs for Revision]
└─$ ./memory12
0,1
1,3
2,5
3,7
4,9
5,11
6,13
7,15
8,17
9,19

┌──(kali㉿kali)-[~/Desktop/C Programs for Revision]
└─$ 
```

17. memory13

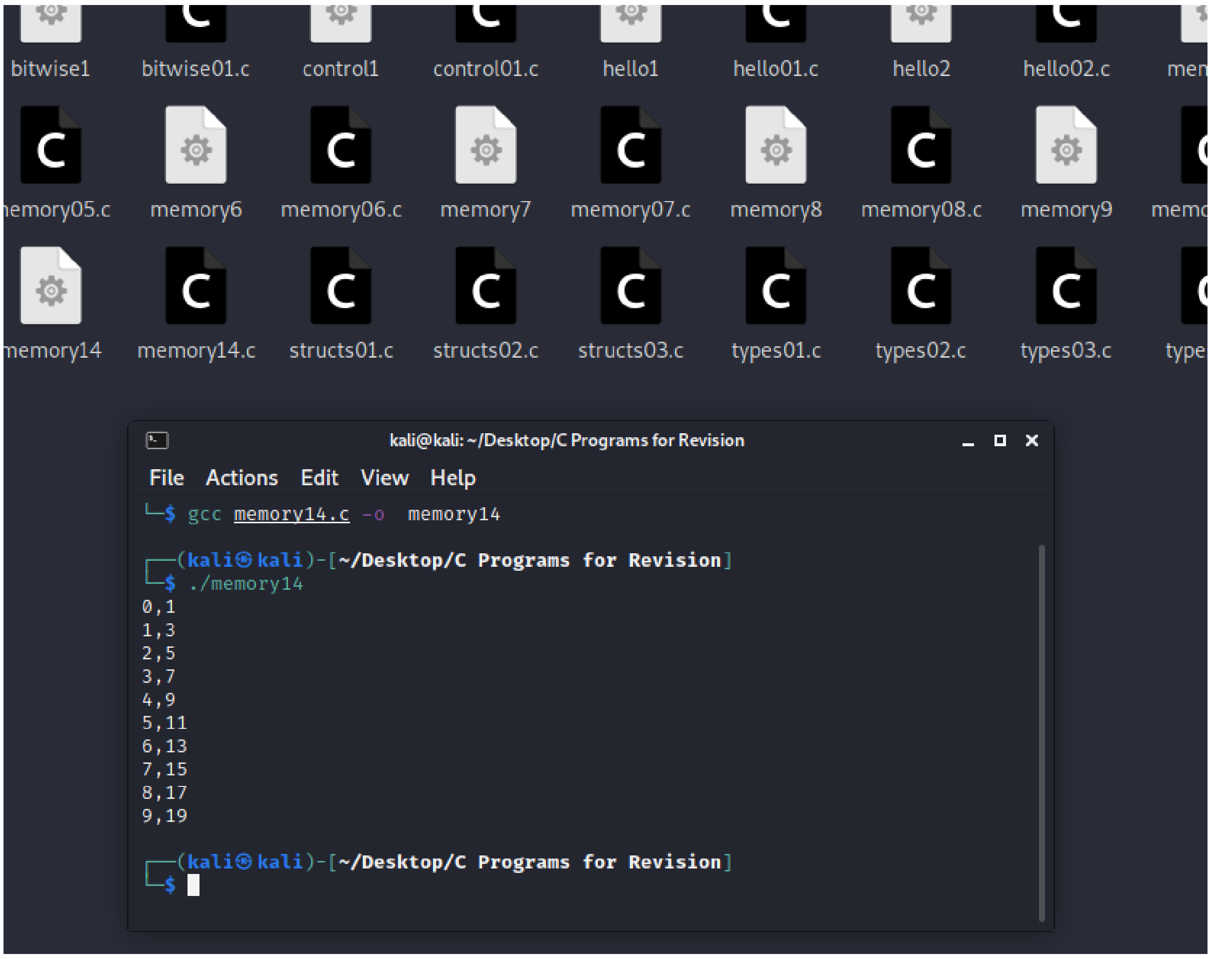
```
#include <stdio.h>
#include <stdlib.h>
#include <malloc.h>

int n = 10;

void inc(int *w) {
  int i;
  for(i=0;i<n;i++){
    *w = *w + 1;
    w++;
  }
}

int main() {
  int i;
  int *x, *y;
  x = malloc(sizeof(int) * n);
  y = x;

  for(i=0;i<n;i++){
    *y = 2 * i;
    y++;
  }

  inc(x);
  y = x;

  for(i=0;i<n;i++){
    printf("%d,%d\n", i, *y);
    y++;
  }
  free(x);
  return EXIT_SUCCESS;
}
```



```
└─$ gcc memory13.c -o  memory13

┌──(kali㉿kali)-[~/Desktop/C Programs for Revision]
└─$ ./memory13
0,1
1,3
2,5
3,7
4,9
5,11
6,13
7,15
8,17
9,19

┌──(kali㉿kali)-[~/Desktop/C Programs for Revision]
└─$ 
```

18. memory14

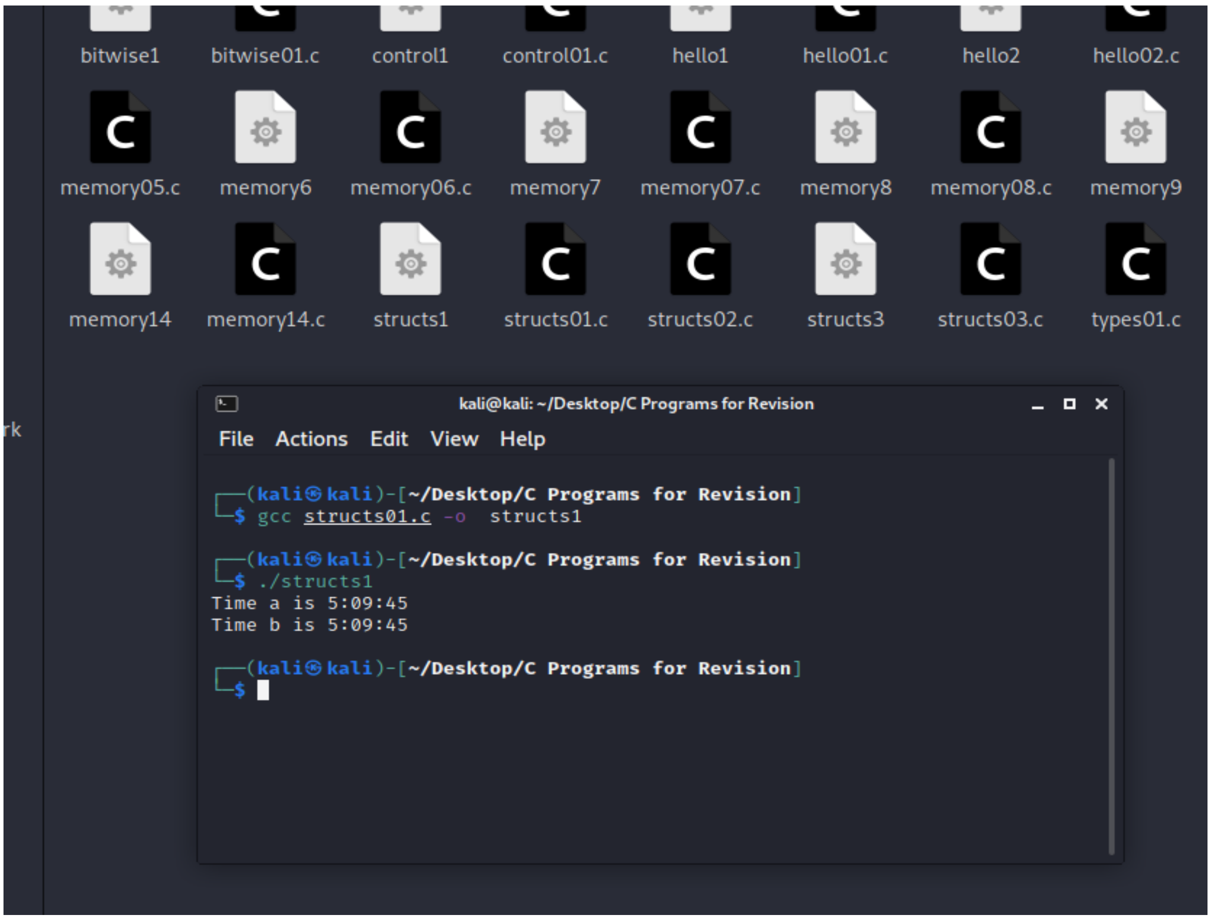
```
#include <stdio.h>
#include <stdlib.h>
#include <malloc.h>

int n = 10;

void initialise(int *w) {
  int i;
  for(i=0;i<n;i++){
    *w = 2 * i;
    w++;
  }
}

void inc(int *w) {
  int i;
  for(i=0;i<n;i++){
    *w = *w + 1;
    w++;
  }
}

void output(int *w) {
  int i;
  for(i=0;i<n;i++){
    printf("%d,%d\n", i, w[i]);
  }
}

int main() {
  int *x;
  x = malloc(sizeof(int) * n);

  initialise(x);
  inc(x);
  output(x);



  free(x);
  return EXIT_SUCCESS;
}
```



```
└─$ gcc memory14.c -o  memory14

┌──(kali㉿kali)-[~/Desktop/C Programs for Revision]
└─$ ./memory14
0,1
1,3
2,5
3,7
4,9
5,11
6,13
7,15
8,17
9,19

┌──(kali㉿kali)-[~/Desktop/C Programs for Revision]
└─$ 
```

19. structs01

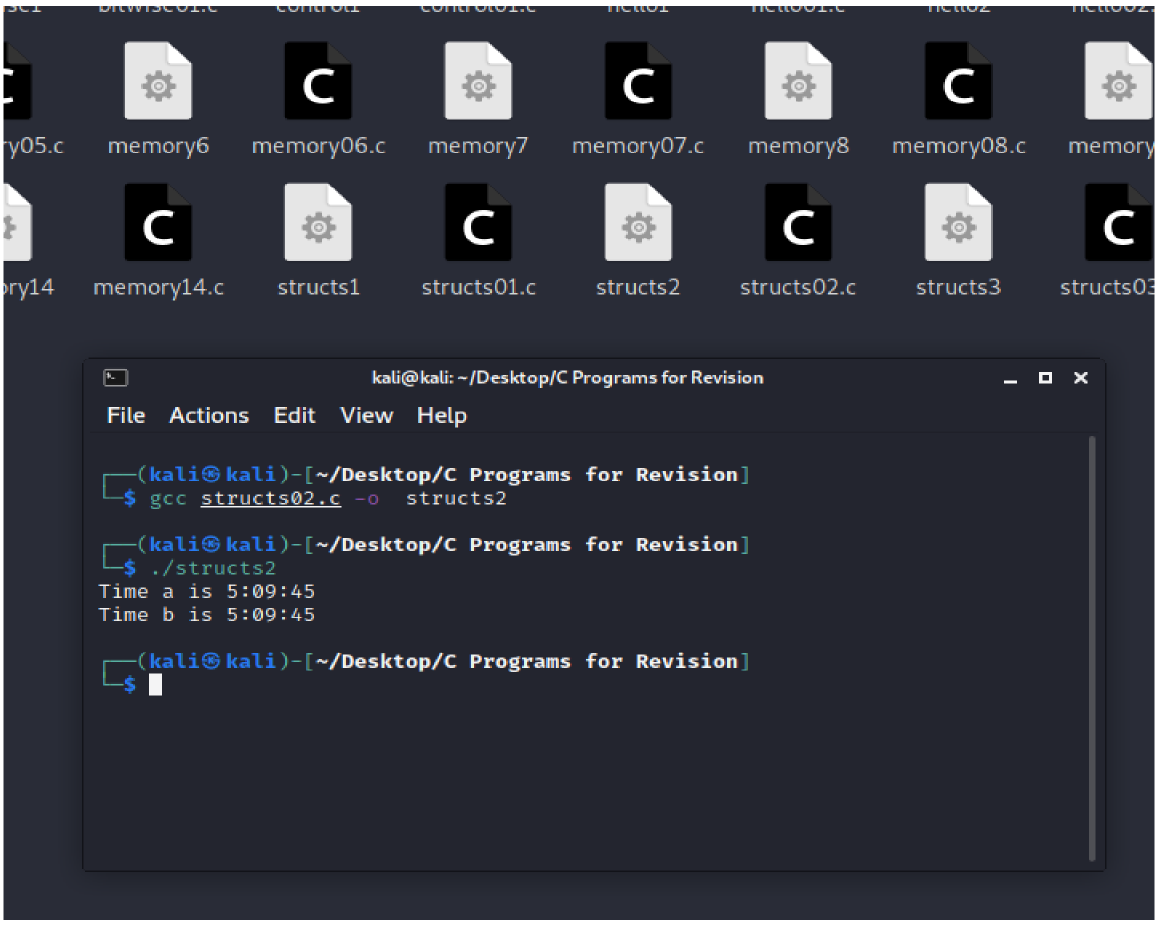
```
#include <stdio.h>
struct t {
  unsigned int h;
  unsigned int m;
  unsigned int s;
};

int main() {
  struct t a;
  struct t *b;

  a.h = 5;
  a.m = 9;
  a.s = 45;

  printf("Time a is %u:%02u:%02u\n", a.h, a.m, a.s);

  b = &a;
  printf("Time b is %u:%02u:%02u\n", b->h, b->m, b->s);

  return 0;
}
```



```
┌──(kali㉿kali)-[~/Desktop/C Programs for Revision]
└─$ gcc structs01.c -o  structs1

┌──(kali㉿kali)-[~/Desktop/C Programs for Revision]
└─$ ./structs1
Time a is 5:09:45
Time b is 5:09:45

┌──(kali㉿kali)-[~/Desktop/C Programs for Revision]
└─$ 
```

20. structs02

```
#include <stdio.h>

typedef struct {
  unsigned int h;
  unsigned int m;
  unsigned int s;
} t;

int main() {
  t a;
  t *b;

  a.h = 5;
  a.m = 9;
  a.s = 45;

  printf("Time a is %u:%02u:%02u\n", a.h, a.m, a.s);

  b = &a;
  printf("Time b is %u:%02u:%02u\n", b->h, b->m, b->s);

  return 0;
}
```

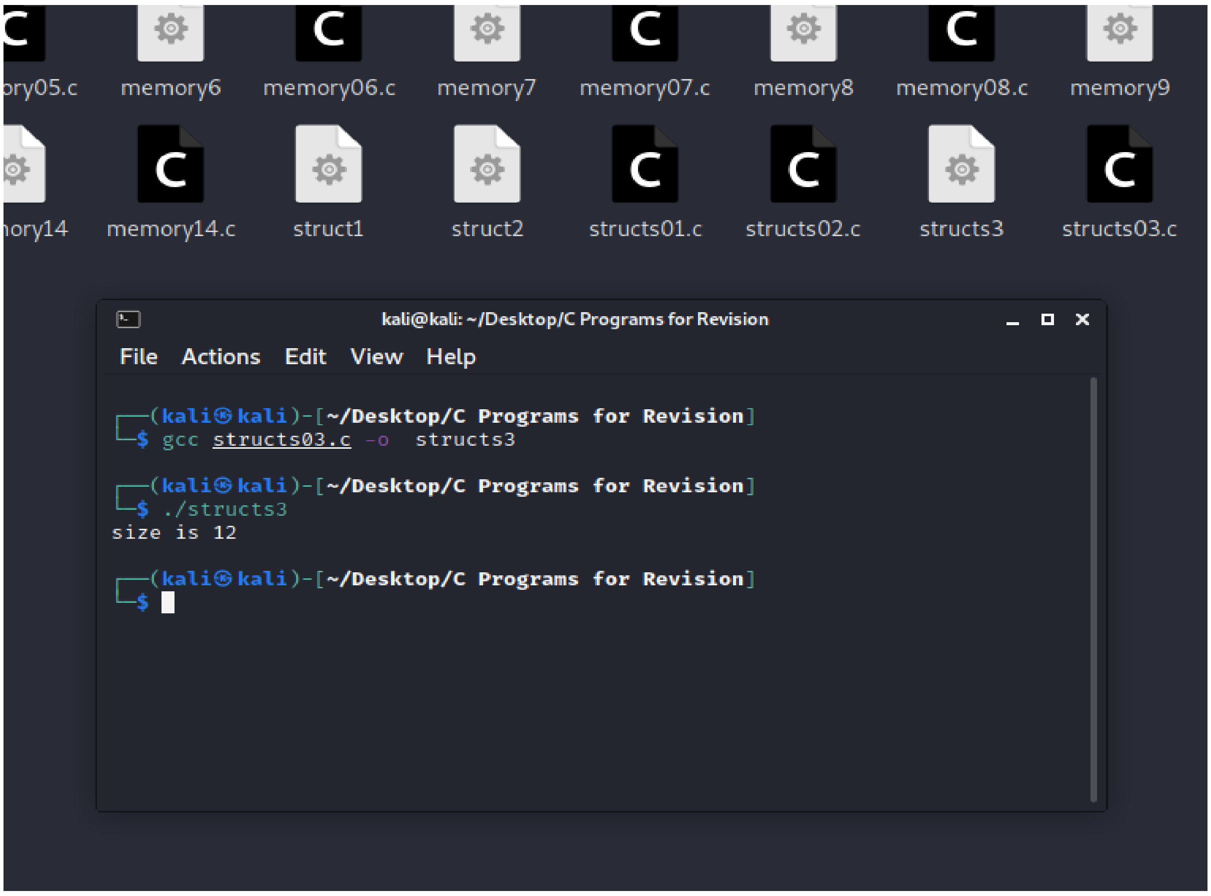


```
┌──(kali㉿kali)-[~/Desktop/C Programs for Revision]
└─$ gcc structs02.c -o  structs2

┌──(kali㉿kali)-[~/Desktop/C Programs for Revision]
└─$ ./structs2
Time a is 5:09:45
Time b is 5:09:45

┌──(kali㉿kali)-[~/Desktop/C Programs for Revision]
└─$ 
```

21. structs03

```
#include <stdio.h>
#include <malloc.h>

typedef struct {
  unsigned int h;
  unsigned int m;
  unsigned int s;
} t;

int main() {
  t *a;

 a = (t *) malloc(sizeof(t));
```

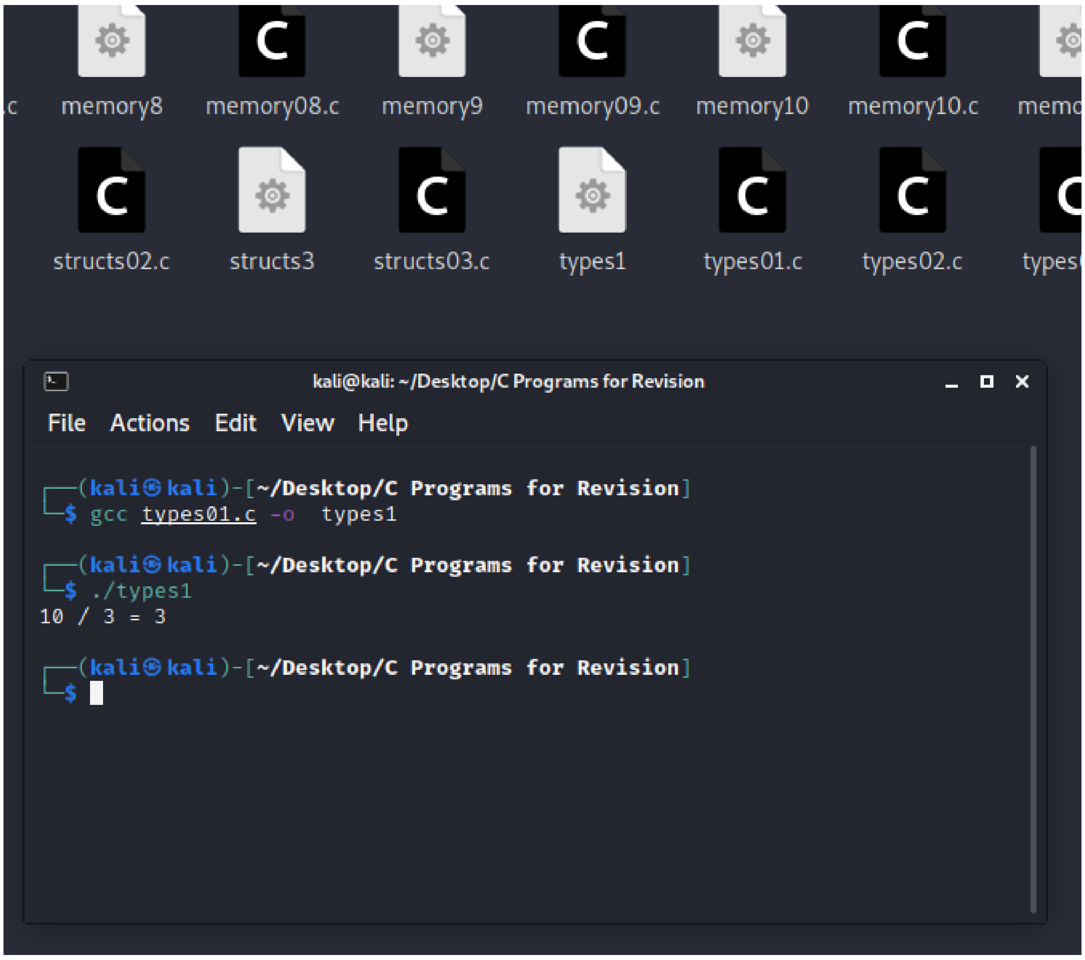
```
  a->h = 5;
  a->m = 9;
  a->s = 45;
  //printf("Time a is %u:%02u:%02u\n", a->h, a->m, a->s);

  printf("size is %d", sizeof(t));

  //free(a);
  return 0;
}
```



```
┌──(kali㉿kali)-[~/Desktop/C Programs for Revision]
└─$ gcc structs03.c -o  structs3

┌──(kali㉿kali)-[~/Desktop/C Programs for Revision]
└─$ ./structs3
size is 12

┌──(kali㉿kali)-[~/Desktop/C Programs for Revision]
└─$ 
```

22. types01

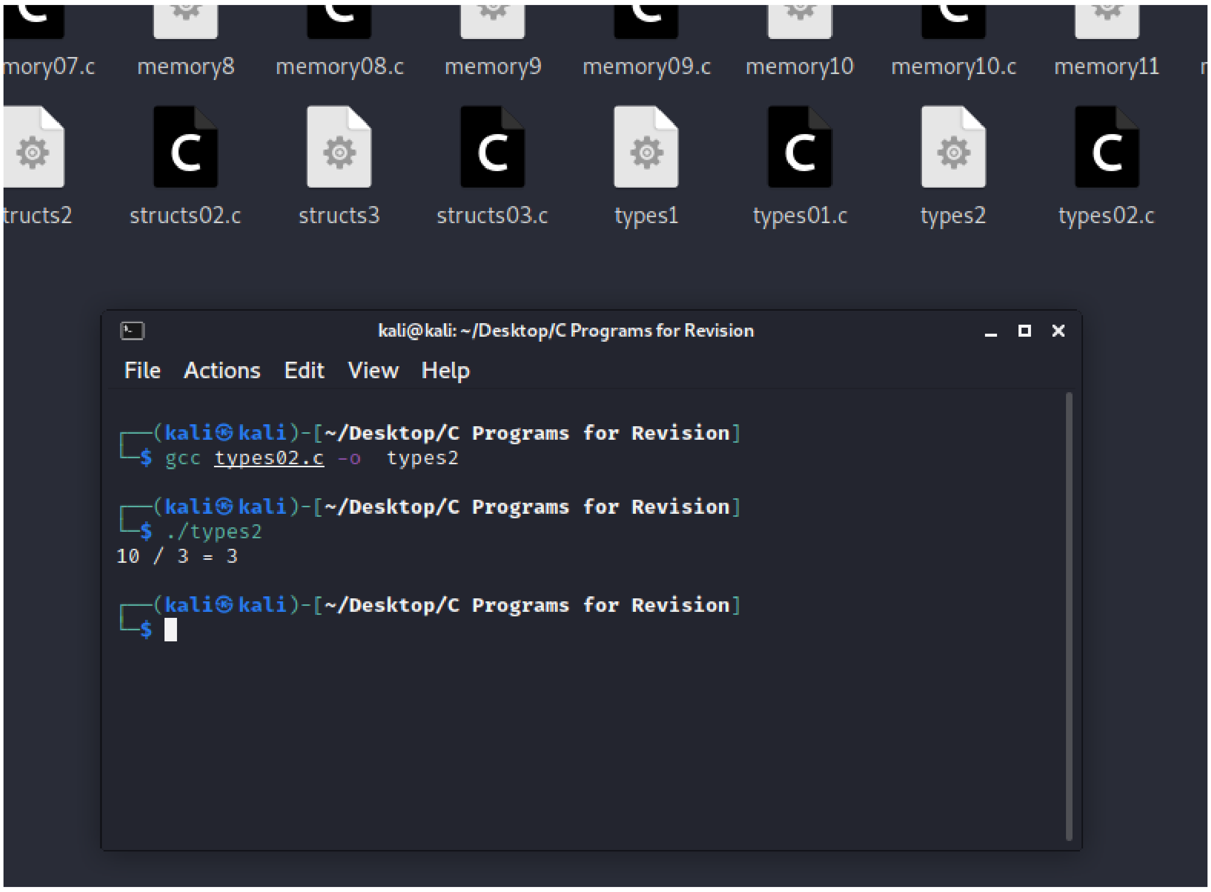
```
#include <stdlib.h>
#include <stdio.h>

int main() {
  int x = 10;
  int y = 3;

  printf("%d / %d = %d\n", x, y, x/y);
  return EXIT_SUCCESS;
}
```



```
┌──(kali㉿kali)-[~/Desktop/C Programs for Revision]
└─$ gcc types01.c -o  types1

┌──(kali㉿kali)-[~/Desktop/C Programs for Revision]
└─$ ./types1
10 / 3 = 3

┌──(kali㉿kali)-[~/Desktop/C Programs for Revision]
└─$ 
```

23. types02

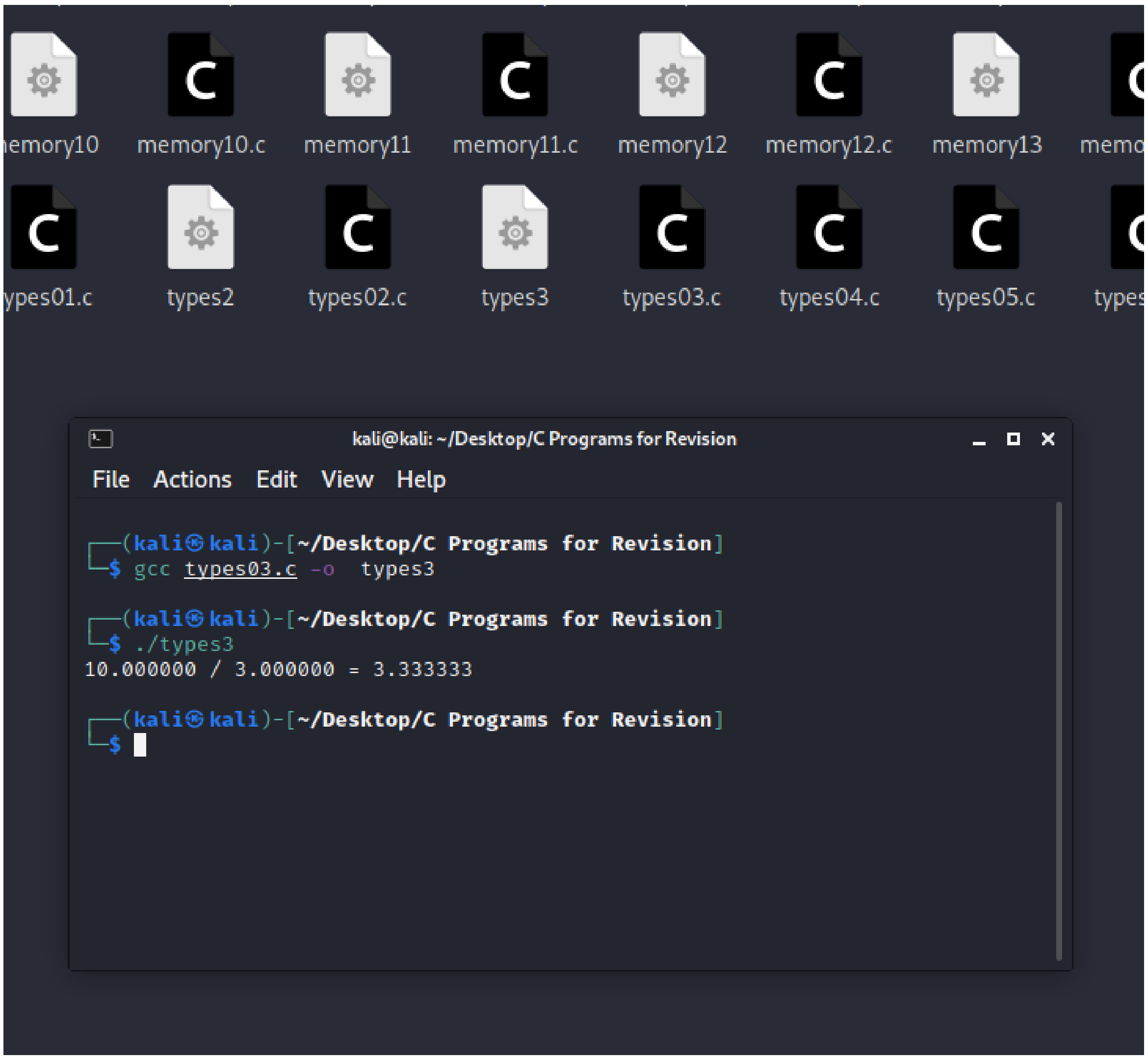
```
#include <stdlib.h>
#include <stdio.h>

int main() {
  long int x = 10L;
  long int y = 3L;
  printf("%ld / %ld = %ld\n", x, y, x/y);
  return EXIT_SUCCESS;
}
```



```
┌──(kali㉿kali)-[~/Desktop/C Programs for Revision]
└─$ gcc types02.c -o  types2

┌──(kali㉿kali)-[~/Desktop/C Programs for Revision]
└─$ ./types2
10 / 3 = 3

┌──(kali㉿kali)-[~/Desktop/C Programs for Revision]
└─$ 
```

24. types03

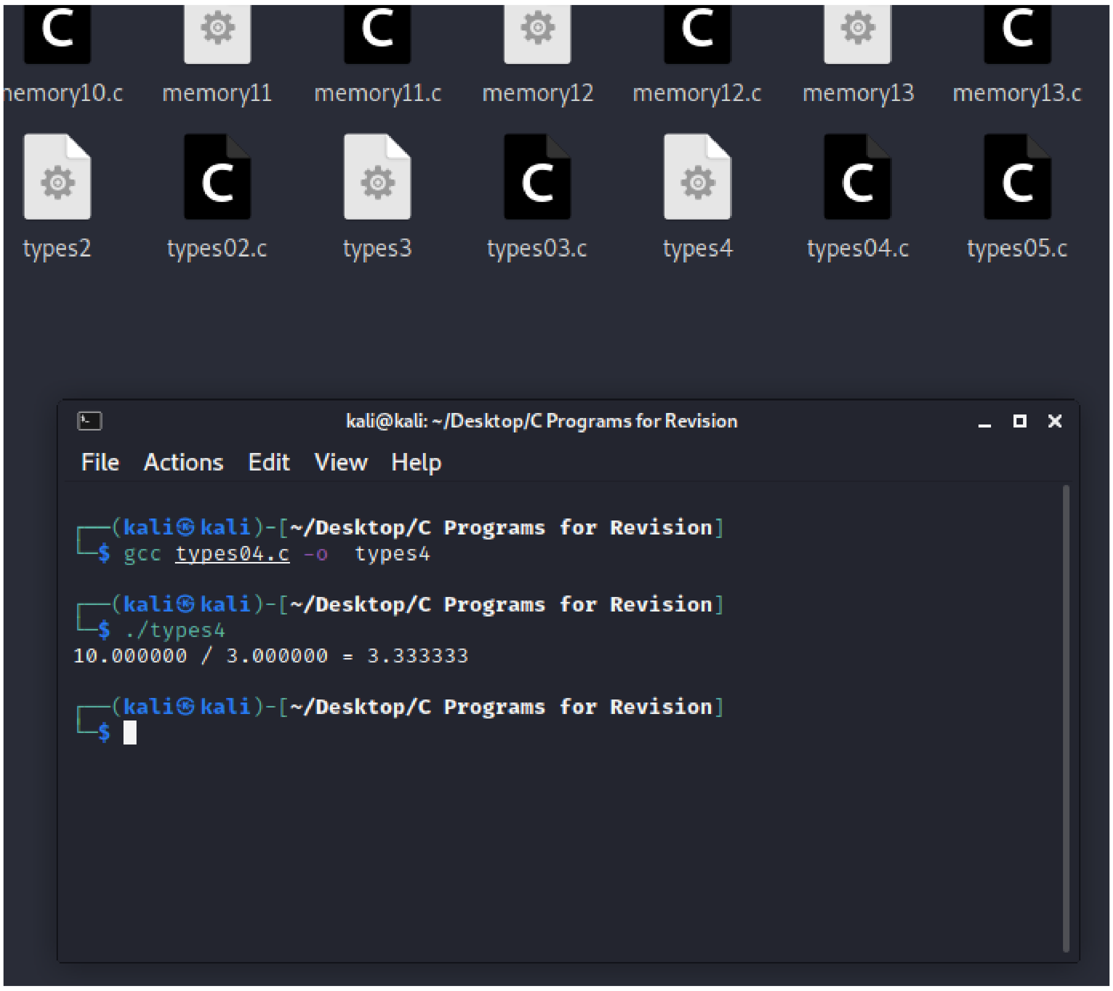
```
#include <stdlib.h>
#include <stdio.h>

int main() {
  float x = 10.0f;
  float y = 3.0f;

  printf("%f / %f = %f\n", x, y, x/y);
  return EXIT_SUCCESS;
}
```



```
┌──(kali㉿kali)-[~/Desktop/C Programs for Revision]
└─$ gcc types03.c -o  types3

┌──(kali㉿kali)-[~/Desktop/C Programs for Revision]
└─$ ./types3
10.000000 / 3.000000 = 3.333333

┌──(kali㉿kali)-[~/Desktop/C Programs for Revision]
└─$ 
```

25. types04

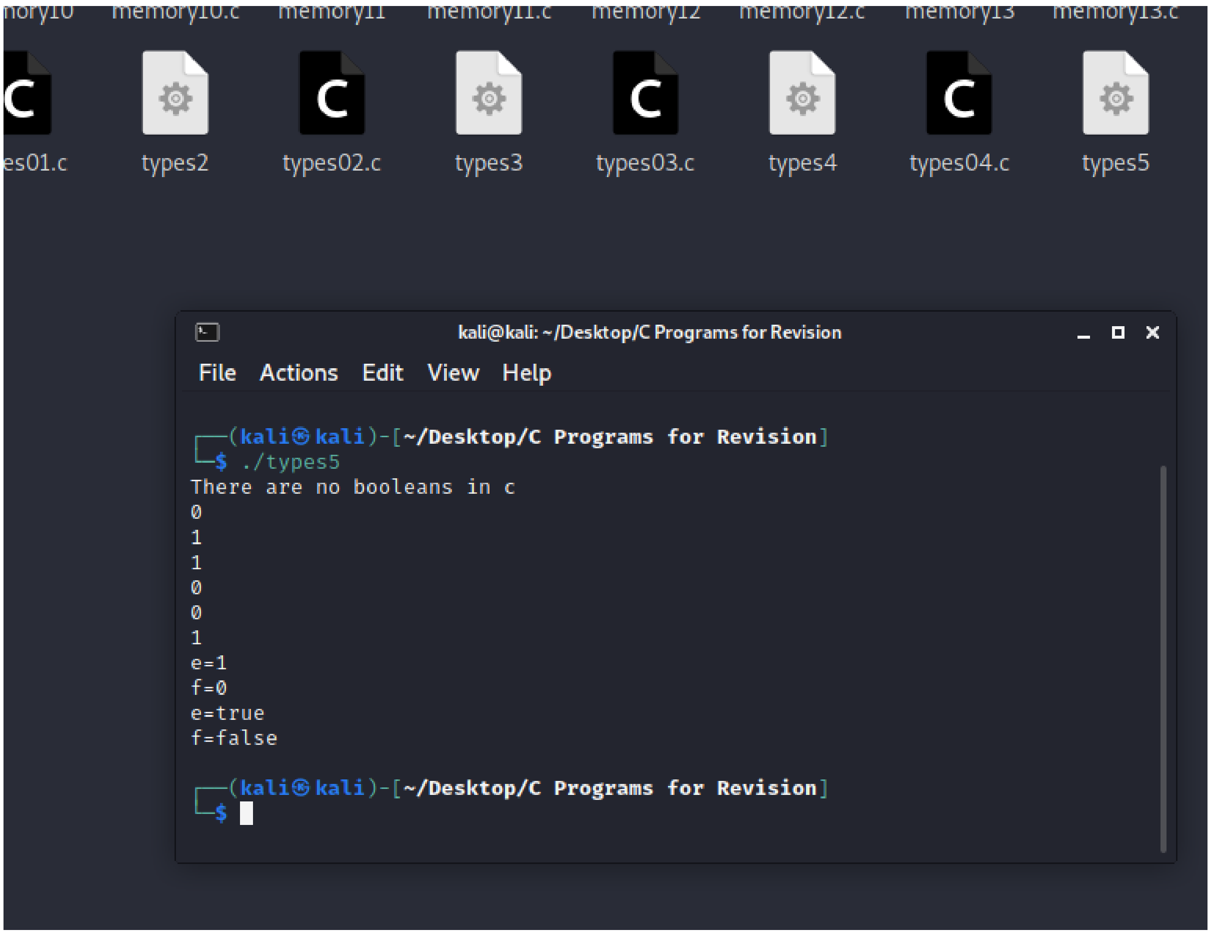
```
#include <stdlib.h>
#include <stdio.h>

int main() {
  double x = 10.0;
  double y = 3.0;

  printf("%lf / %lf = %lf\n", x, y, x/y);
  return EXIT_SUCCESS;
}
```



```
┌──(kali㉿kali)-[~/Desktop/C Programs for Revision]
└─$ gcc types04.c -o  types4

┌──(kali㉿kali)-[~/Desktop/C Programs for Revision]
└─$ ./types4
10.000000 / 3.000000 = 3.333333

┌──(kali㉿kali)-[~/Desktop/C Programs for Revision]
└─$ 
```

26. types05

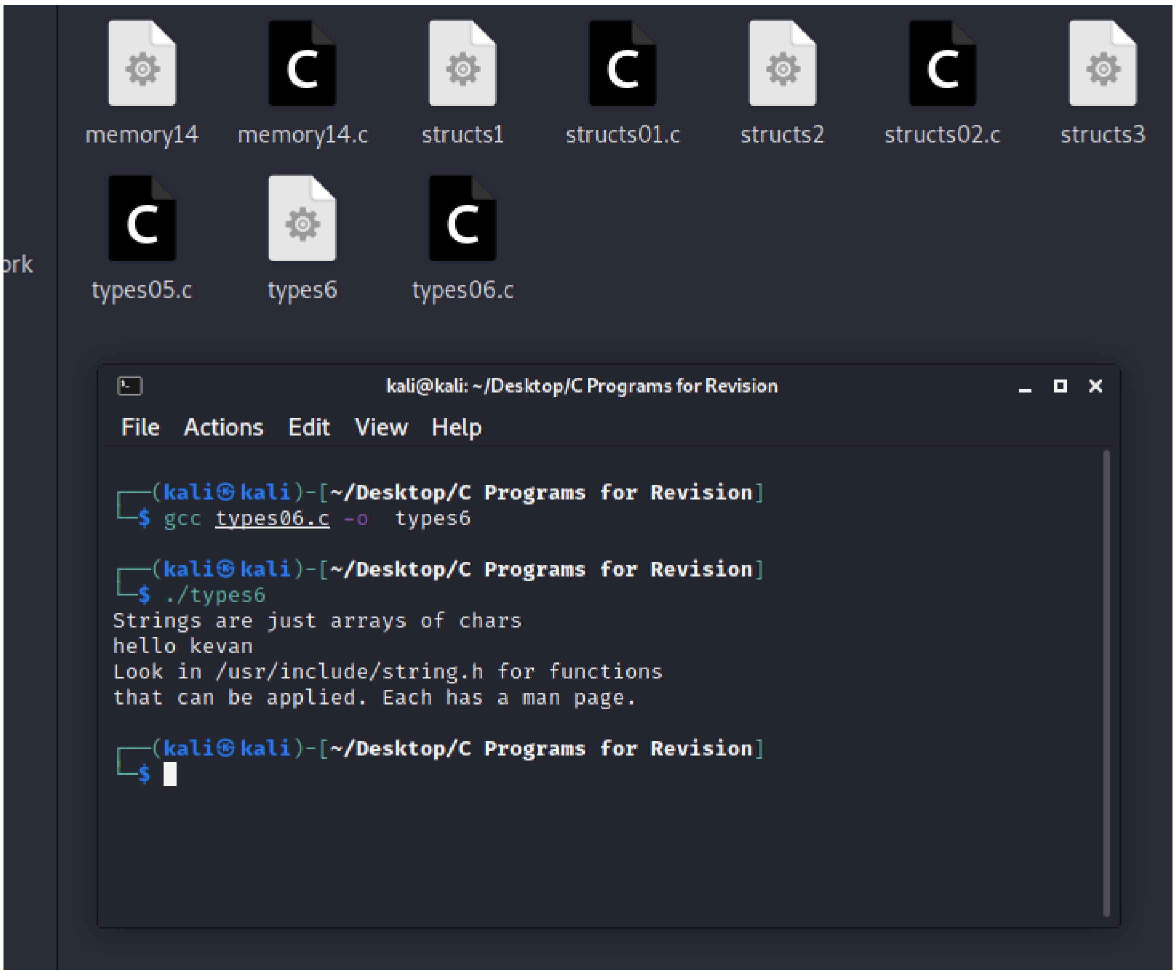
```
#include <stdlib.h>
#include <stdio.h>

int main() {
  int a = 2;
  int b = 3;
  int c = 2;
  int d = 4;

  printf("There are no booleans in c\n");
  printf("%d\n", a==b);
  printf("%d\n", a==c);
  printf("%d\n", a!=b);
  printf("%d\n", a!=c);
  printf("%d\n", a==b);
  printf("%d\n", !(a==b));

  int e = (a == b) || (a == c);
  int f = (a == b) && (a == c);

  printf("e=%d\n", e);
  printf("f=%d\n", f);

  if(e) {
    printf("e=true\n");
  } else {
    printf("e=false\n");
  }

  if(f) {
    printf("f=true\n");
  } else {
    printf("f=false\n");
  }

  return EXIT_SUCCESS;
}
```



```
┌──(kali㉿kali)-[~/Desktop/C Programs for Revision]
└─$ ./types5
There are no booleans in c
0
1
1
0
0
1
e=1
f=0
e=true
f=false

┌──(kali㉿kali)-[~/Desktop/C Programs for Revision]
└─$ 
```

27. types06

```
#include <stdlib.h>
#include <stdio.h>

int main() {
  printf("Strings are just arrays of chars\n");

  char *message1 = "hello";
  char *message2 = "kevan";

  printf("%s %s\n", message1, message2);
  printf("Look in /usr/include/string.h for functions\n");
  printf("that can be applied. Each has a man page.\n");
  return EXIT_SUCCESS;
}
```



```
┌──(kali㉿kali)-[~/Desktop/C Programs for Revision]
└─$ gcc types06.c -o  types6

┌──(kali㉿kali)-[~/Desktop/C Programs for Revision]
└─$ ./types6
Strings are just arrays of chars
hello kevan
Look in /usr/include/string.h for functions
that can be applied. Each has a man page.

┌──(kali㉿kali)-[~/Desktop/C Programs for Revision]
└─$ 
```