

Exercise 2

Kendrick Kwong

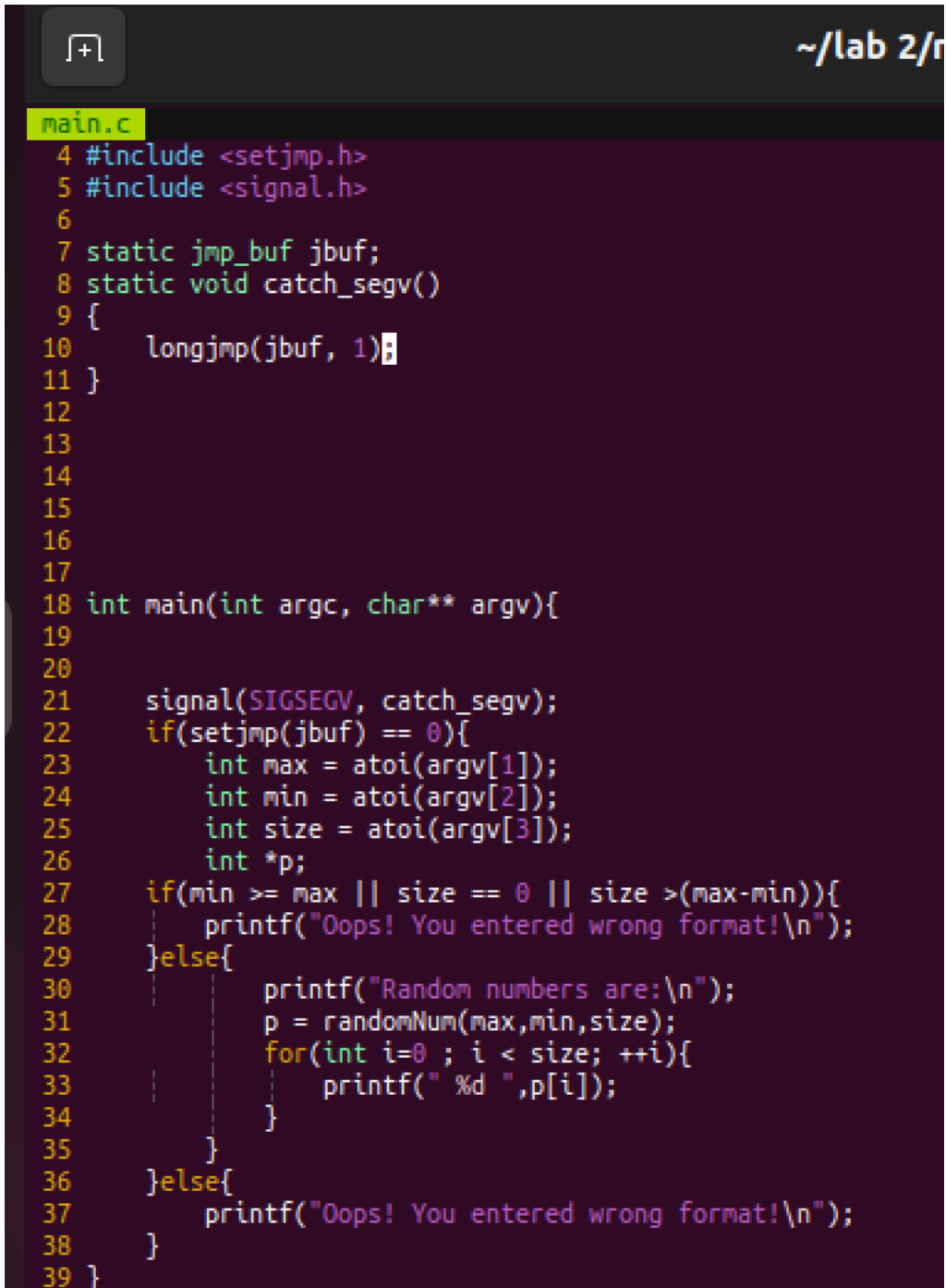
randomNum.h:

```
randomNum.h
1 #ifndef RANDOMNUM_H
2 #define RANDOMNUM_H
3
4 #include <time.h>
5 #include <stdlib.h>
6 #include <stdio.h>
7
8 int * randomNum(int max, int min, int size);
9
10 #endif
```

randomNum.c

```
randomNum.c
1
2 #include <time.h>
3 #include <stdlib.h>
4 #include <stdio.h>
5 #include "randomNum.h"
6 #include <malloc.h>
7
8 int * randomNum(int max, int min, int size){
9     srand(time(NULL));
10    int *r = malloc(sizeof(size));
11
12    for(int i=0 ; i<size ; ++i){
13
14        r[i] = (rand() % (max - min)) + min;
15    }
16
17    free(r);
18    return r;
19 }
20
```

Main.c:



The image shows a terminal window with a dark background. At the top left, there is a button with a plus sign and a square icon. At the top right, the text '~ /lab 2/r' is displayed. Below the header, the filename 'main.c' is highlighted in yellow. The code is written in a monospaced font with syntax highlighting: preprocessor directives and keywords are in blue, identifiers and literals are in green, and string literals are in red. Line numbers 4 through 39 are listed on the left side of the code block.

```
4 #include <setjmp.h>
5 #include <signal.h>
6
7 static jmp_buf jbuf;
8 static void catch_segv()
9 {
10     longjmp(jbuf, 1);
11 }
12
13
14
15
16
17
18 int main(int argc, char** argv){
19
20
21     signal(SIGSEGV, catch_segv);
22     if(setjmp(jbuf) == 0){
23         int max = atoi(argv[1]);
24         int min = atoi(argv[2]);
25         int size = atoi(argv[3]);
26         int *p;
27         if(min >= max || size == 0 || size > (max-min)){
28             printf("Oops! You entered wrong format!\n");
29         }else{
30             printf("Random numbers are:\n");
31             p = randomNum(max,min,size);
32             for(int i=0 ; i < size; ++i){
33                 printf(" %d ",p[i]);
34             }
35         }
36     }else{
37         printf("Oops! You entered wrong format!\n");
38     }
39 }
```

Function test:

```
kendrick807@kendrick807-virtual-machine > ./main 3 2 4
Oops! You entered wrong format!
kendrick807@kendrick807-virtual-machine > ./main 3 2 1
Random numbers are:
2
kendrick807@kendrick807-virtual-machine > ./main 3 2 0
Oops! You entered wrong format!
kendrick807@kendrick807-virtual-machine > ./main 10 2 1
Random numbers are:
9
kendrick807@kendrick807-virtual-machine > ./main 10 2 5
Random numbers are:
2 6 8 4 4
kendrick807@kendrick807-virtual-machine > ./main 2 10 5
Oops! You entered wrong format!
kendrick807@kendrick807-virtual-machine > ./main
Oops! You entered wrong format!
kendrick807@kendrick807-virtual-machine > ./main 10 2
Oops! You entered wrong format!
```

Preprocessor:

To invoke GCC and only carry out the preprocessor command:

```
kendrick807@kendrick807-virtual-machine > gcc -E randomNum.c main.c
# 0 "randomNum.c"
# 0 "<built-in>"
```

`gcc -E in.c -o in.i`

copy the output of the preprocessor to another .c file

The difference between my .c files and what they look like after the preprocessor has done its thing is that it will generate an object file (i.e. _randomNum.o , main.o)

Compilation: