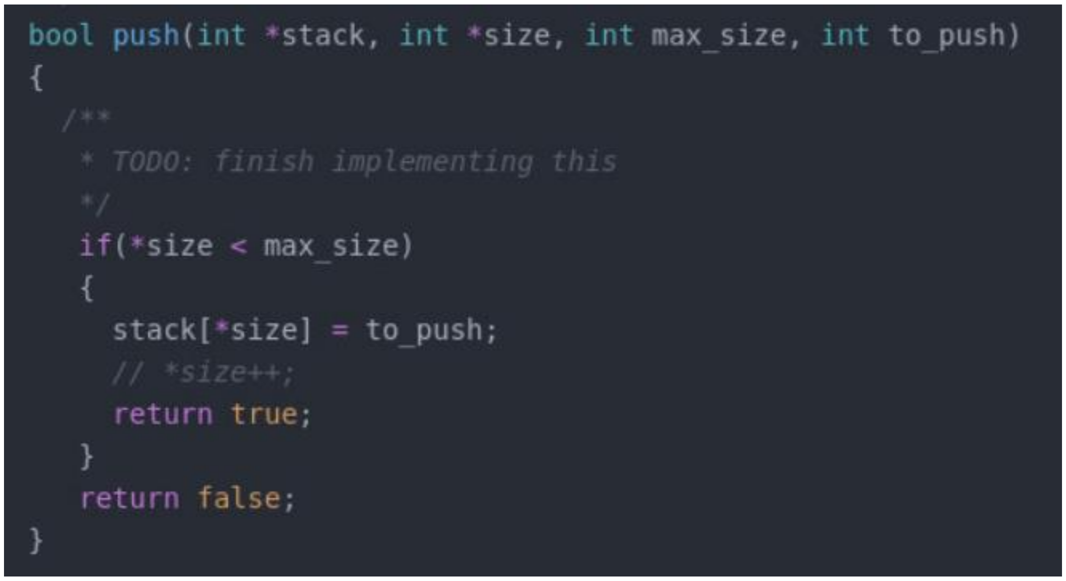
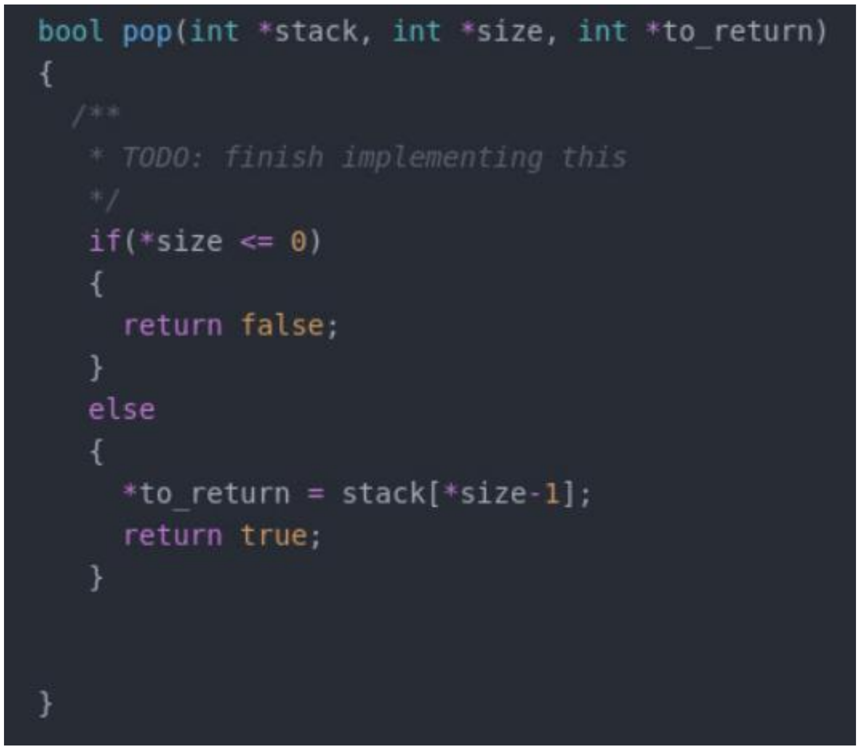
Assignment 2

Push Code



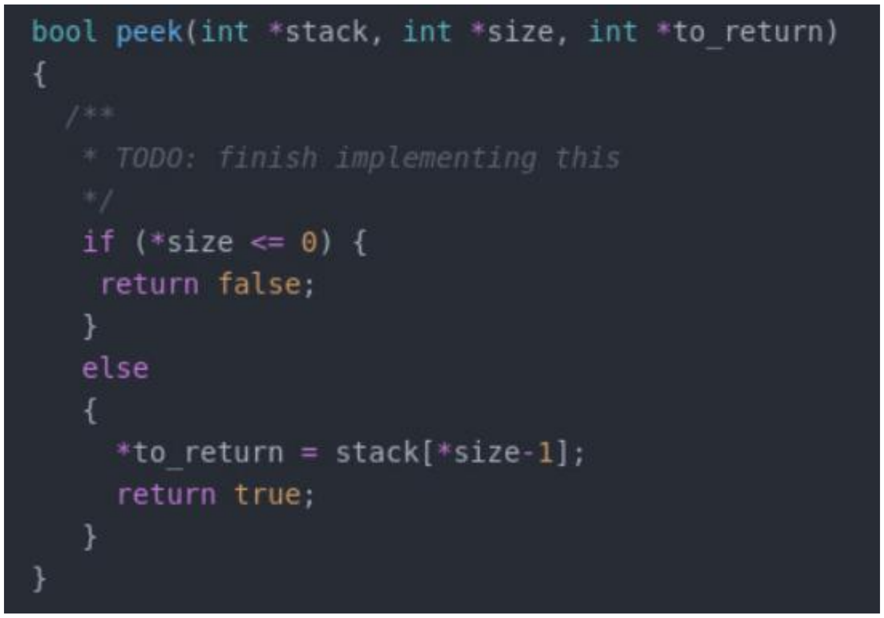
```
bool push(int *stack, int *size, int max_size, int to_push)
{
  /**
   * TODO: finish implementing this
   */
   if(*size < max_size)
   {
     stack[*size] = to_push;
     // *size++;
     return true;
   }
   return false;
}
```

Pop code



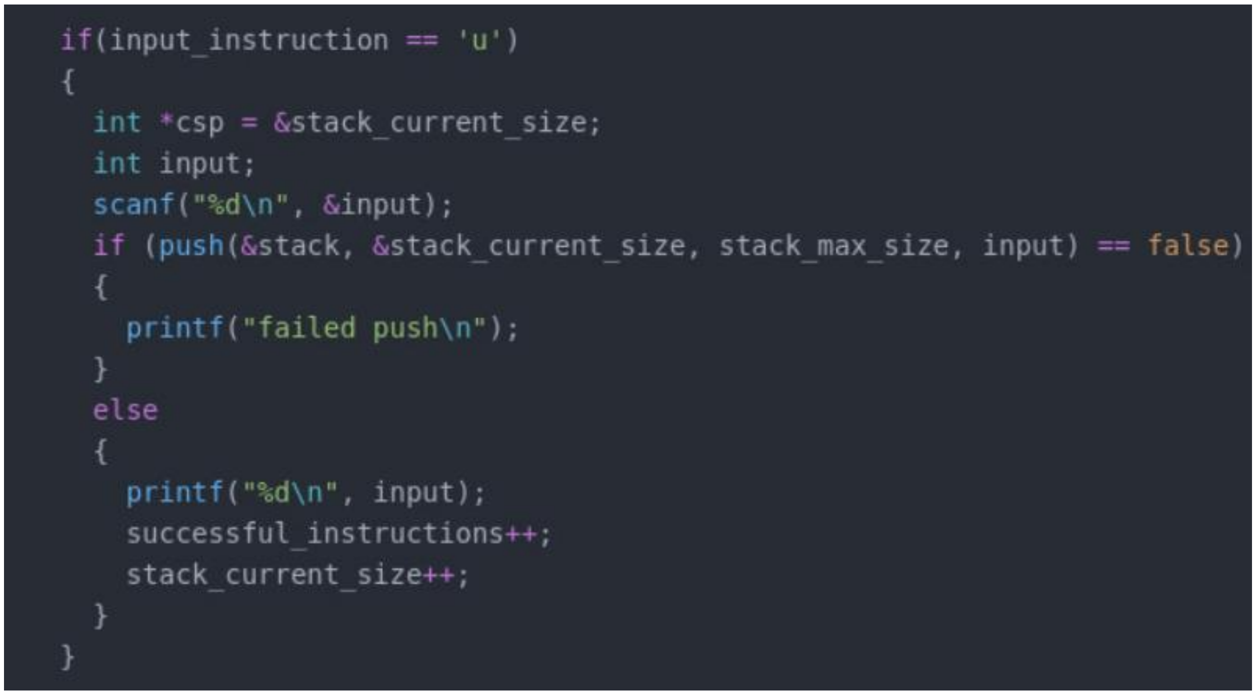
```
bool pop(int *stack, int *size, int *to_return)
{
  /**
   * TODO: finish implementing this
   */
   if(*size <= 0)
   {
     return false;
   }
   else
   {
     *to_return = stack[*size-1];
     return true;
   }


}
```

Peek code

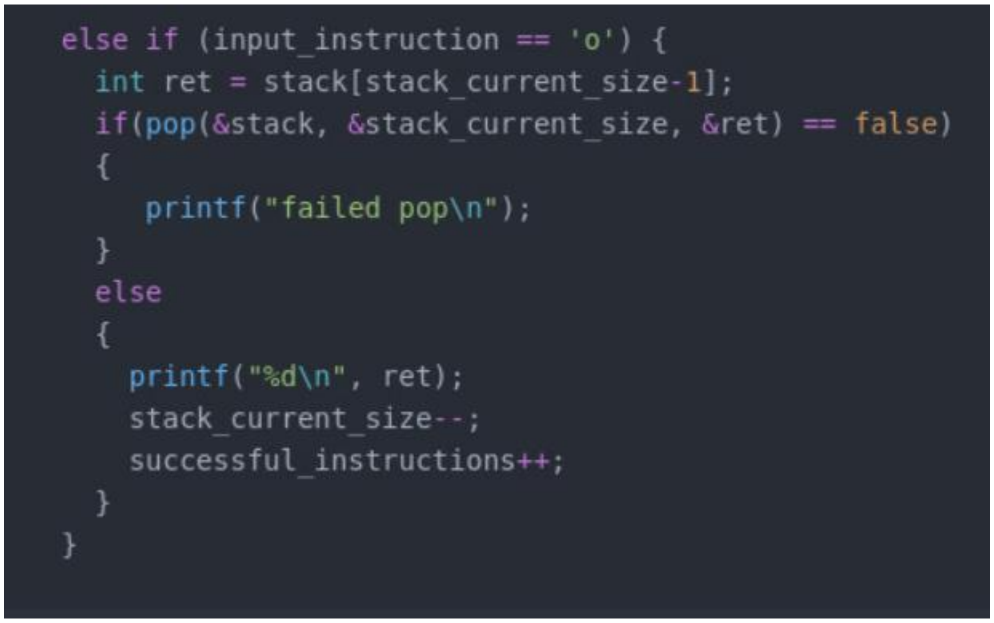


```
bool peek(int *stack, int *size, int *to_return)
{
  /**
   * TODO: finish implementing this
   */
   if (*size <= 0) {
    return false;
   }
   else
   {
     *to_return = stack[*size-1];
     return true;
   }
}
```

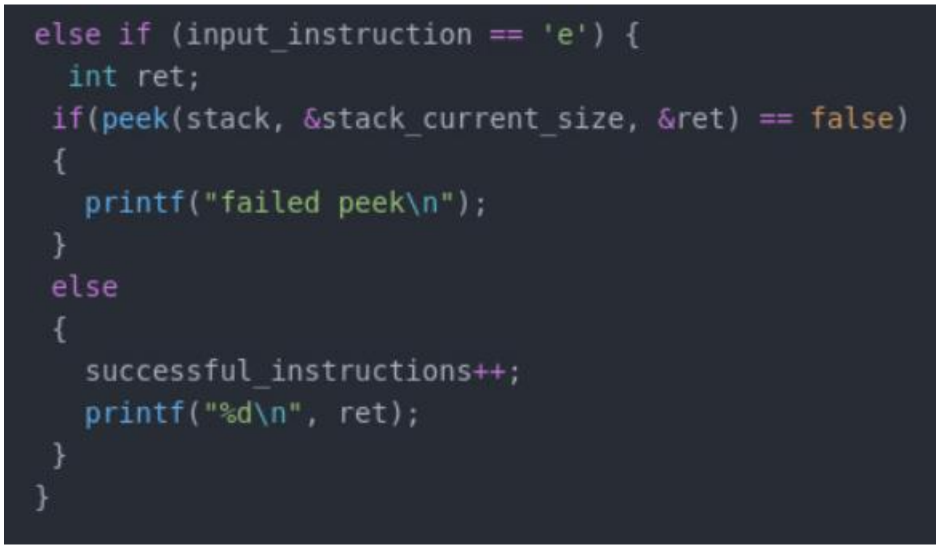
TODO Main code



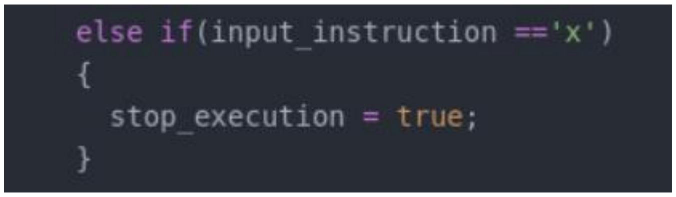
```c
if(input_instruction == 'u')
{
  int *csp = &stack_current_size;
  int input;
  scanf("%d\n", &input);
  if (push(&stack, &stack_current_size, stack_max_size, input) == false)
  {
    printf("failed push\n");
  }
  else
  {
    printf("%d\n", input);
    successful_instructions++;
    stack_current_size++;
  }
}
```



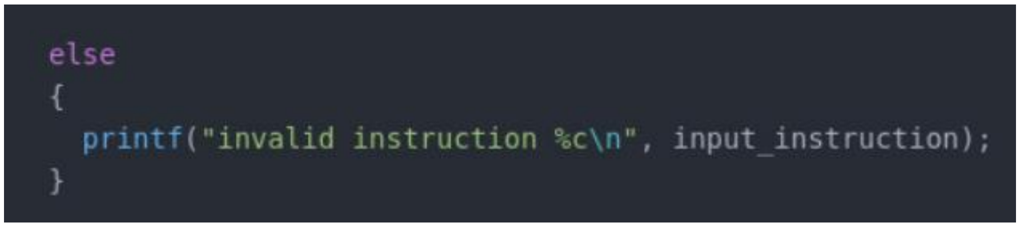
```c
else if (input_instruction == 'o') {
  int ret = stack[stack_current_size-1];
  if(pop(&stack, &stack_current_size, &ret) == false)
  {
     printf("failed pop\n");
  }
  else
  {
    printf("%d\n", ret);
    stack_current_size--;
    successful_instructions++;
  }
}
```



```c
else if (input_instruction == 'e') {
  int ret;
 if(peek(stack, &stack_current_size, &ret) == false)
 {
   printf("failed peek\n");
 }
 else
 {
   successful_instructions++;
   printf("%d\n", ret);
 }
}
```

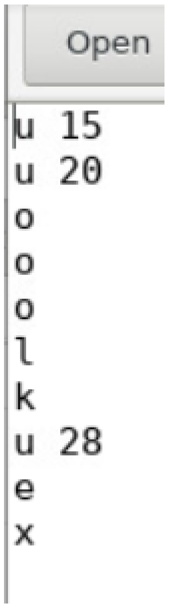


```c
  else if(input_instruction =='x')
  {
    stop_execution = true;
  }
```



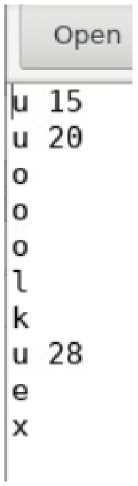
```c
 else
 {
   printf("invalid instruction %c\n", input_instruction);
 }
```

My test case input



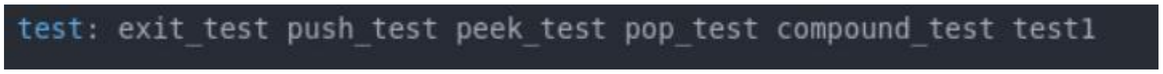
```
u 15
u 20
o
o
o
l
k
u 28
e
x
```

My test case expected output

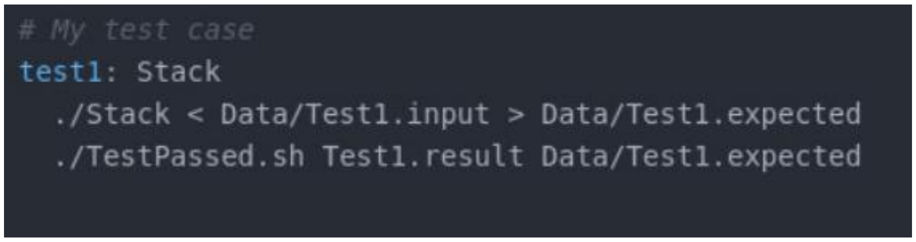


```
u 15
u 20
o
o
o
l
k
u 28
e
x
```

Makefile modification to include my testcase



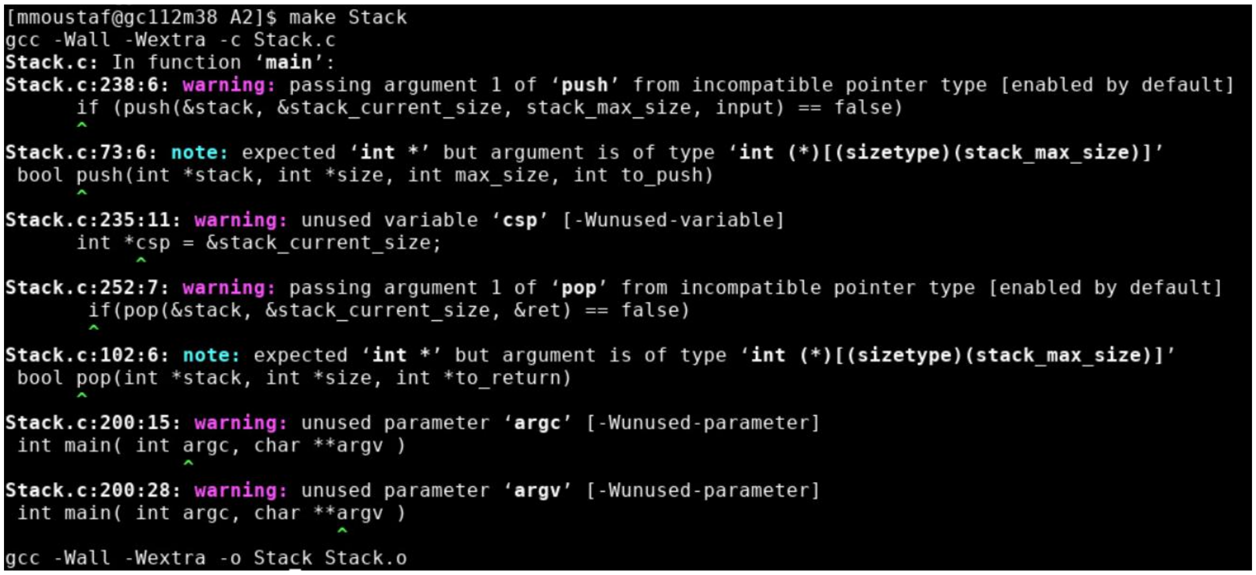
```
test: exit_test push_test peek_test pop_test compound_test test1
```



```
# My test case
test1: Stack
  ./Stack < Data/Test1.input > Data/Test1.expected
  ./TestPassed.sh Test1.result Data/Test1.expected
```

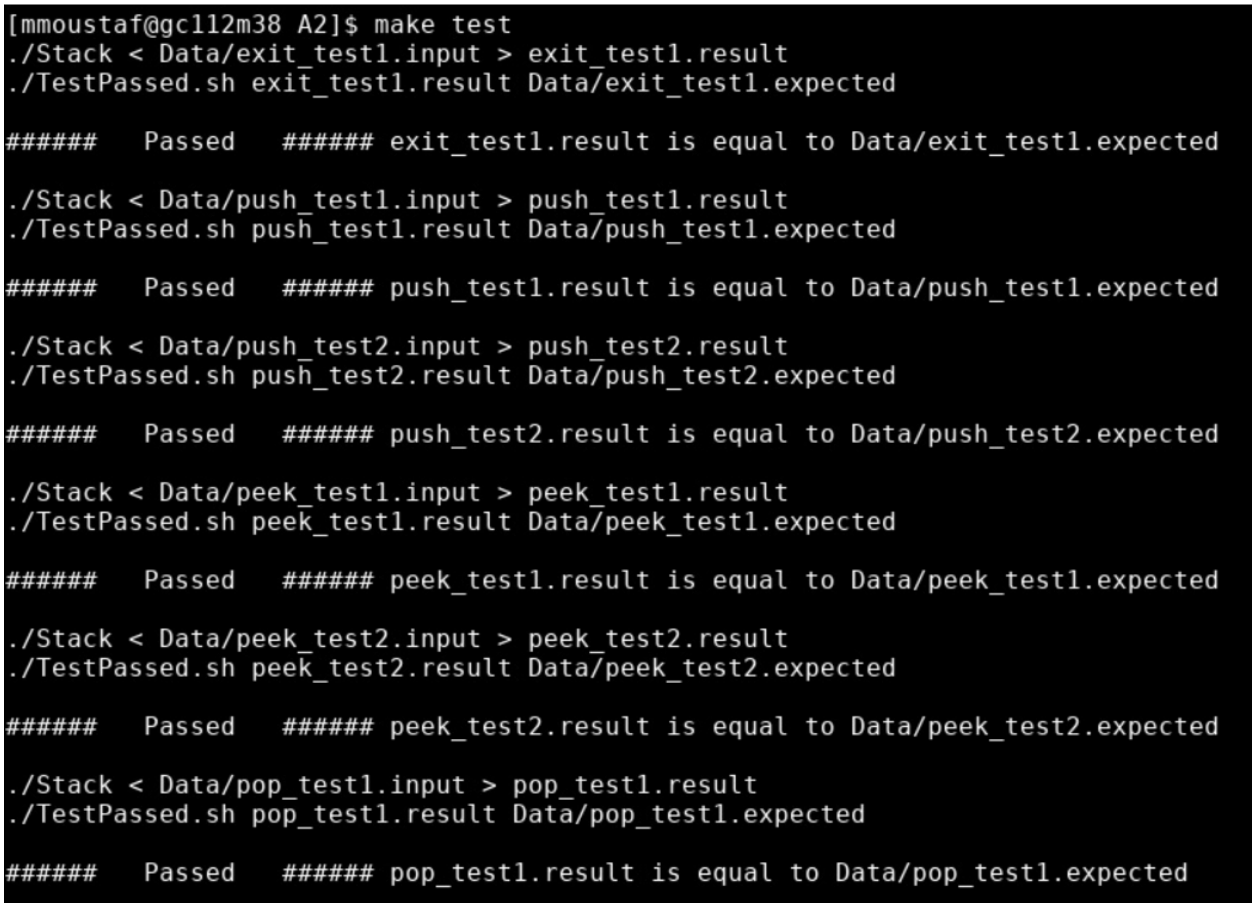
Terminal Activity

Make Stack

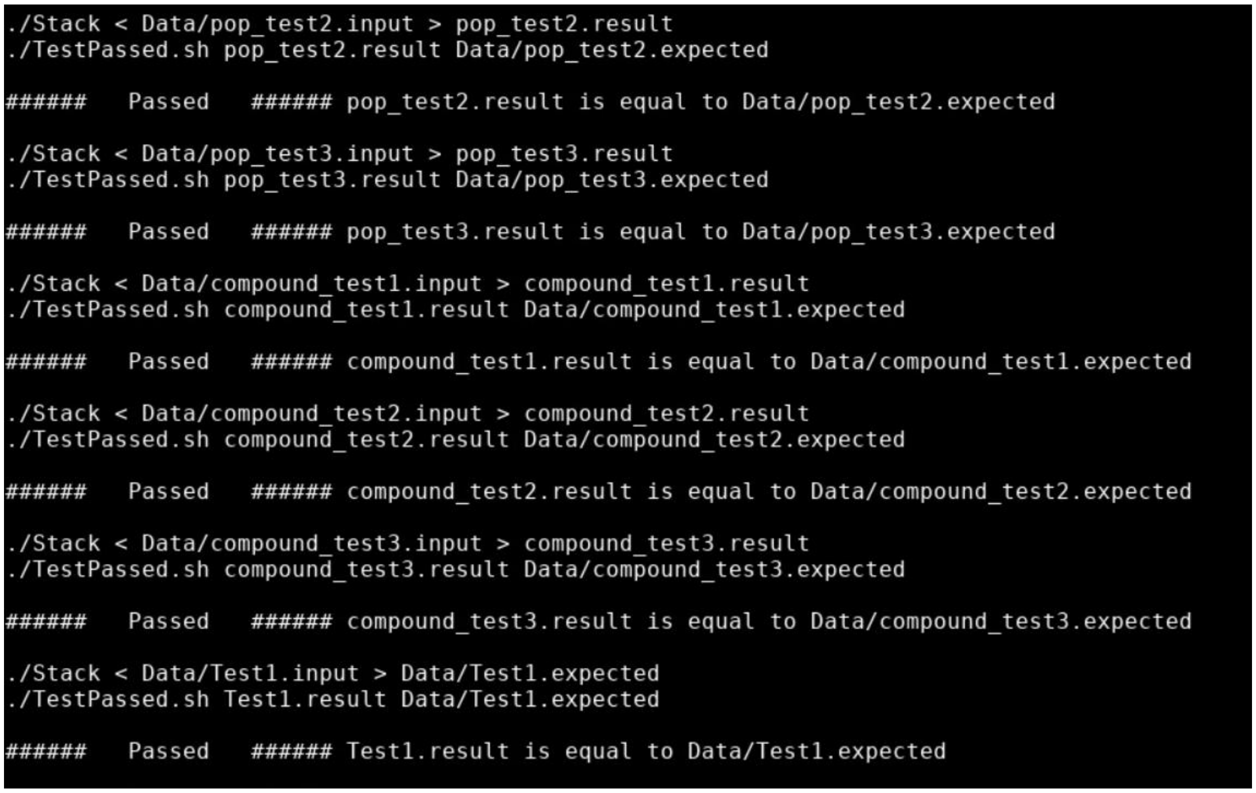


```
[mmoustaf@gc112m38 A2]$ make Stack
gcc -Wall -Wextra -c Stack.c
Stack.c: In function ‘main’:
Stack.c:238:6: warning: passing argument 1 of ‘push’ from incompatible pointer type [enabled by default]
      if (push(&stack, &stack_current_size, stack_max_size, input) == false)
      ^
Stack.c:73:6: note: expected ‘int *’ but argument is of type ‘int (*)[(sizetype)(stack_max_size)]’
 bool push(int *stack, int *size, int max_size, int to_push)
      ^
Stack.c:235:11: warning: unused variable ‘csp’ [-Wunused-variable]
      int *csp = &stack_current_size;
           ^
Stack.c:252:7: warning: passing argument 1 of ‘pop’ from incompatible pointer type [enabled by default]
       if(pop(&stack, &stack_current_size, &ret) == false)
       ^
Stack.c:102:6: note: expected ‘int *’ but argument is of type ‘int (*)[(sizetype)(stack_max_size)]’
 bool pop(int *stack, int *size, int *to_return)
      ^
Stack.c:200:15: warning: unused parameter ‘argc’ [-Wunused-parameter]
 int main( int argc, char **argv )
               ^
Stack.c:200:28: warning: unused parameter ‘argv’ [-Wunused-parameter]
 int main( int argc, char **argv )
                            ^
gcc -Wall -Wextra -o Stack Stack.o
```

Make test



```
[mmoustaf@gc112m38 A2]$ make test
./Stack < Data/exit_test1.input > exit_test1.result
./TestPassed.sh exit_test1.result Data/exit_test1.expected

######   Passed   ###### exit_test1.result is equal to Data/exit_test1.expected

./Stack < Data/push_test1.input > push_test1.result
./TestPassed.sh push_test1.result Data/push_test1.expected

######   Passed   ###### push_test1.result is equal to Data/push_test1.expected

./Stack < Data/push_test2.input > push_test2.result
./TestPassed.sh push_test2.result Data/push_test2.expected

######   Passed   ###### push_test2.result is equal to Data/push_test2.expected

./Stack < Data/peek_test1.input > peek_test1.result
./TestPassed.sh peek_test1.result Data/peek_test1.expected

######   Passed   ###### peek_test1.result is equal to Data/peek_test1.expected

./Stack < Data/peek_test2.input > peek_test2.result
./TestPassed.sh peek_test2.result Data/peek_test2.expected

######   Passed   ###### peek_test2.result is equal to Data/peek_test2.expected

./Stack < Data/pop_test1.input > pop_test1.result
./TestPassed.sh pop_test1.result Data/pop_test1.expected

######   Passed   ###### pop_test1.result is equal to Data/pop_test1.expected
```



```
./Stack < Data/pop_test2.input > pop_test2.result
./TestPassed.sh pop_test2.result Data/pop_test2.expected

######   Passed   ###### pop_test2.result is equal to Data/pop_test2.expected

./Stack < Data/pop_test3.input > pop_test3.result
./TestPassed.sh pop_test3.result Data/pop_test3.expected

######   Passed   ###### pop_test3.result is equal to Data/pop_test3.expected

./Stack < Data/compound_test1.input > compound_test1.result
./TestPassed.sh compound_test1.result Data/compound_test1.expected

######   Passed   ###### compound_test1.result is equal to Data/compound_test1.expected

./Stack < Data/compound_test2.input > compound_test2.result
./TestPassed.sh compound_test2.result Data/compound_test2.expected

######   Passed   ###### compound_test2.result is equal to Data/compound_test2.expected

./Stack < Data/compound_test3.input > compound_test3.result
./TestPassed.sh compound_test3.result Data/compound_test3.expected

######   Passed   ###### compound_test3.result is equal to Data/compound_test3.expected

./Stack < Data/Test1.input > Data/Test1.expected
./TestPassed.sh Test1.result Data/Test1.expected

######   Passed   ###### Test1.result is equal to Data/Test1.expected
```