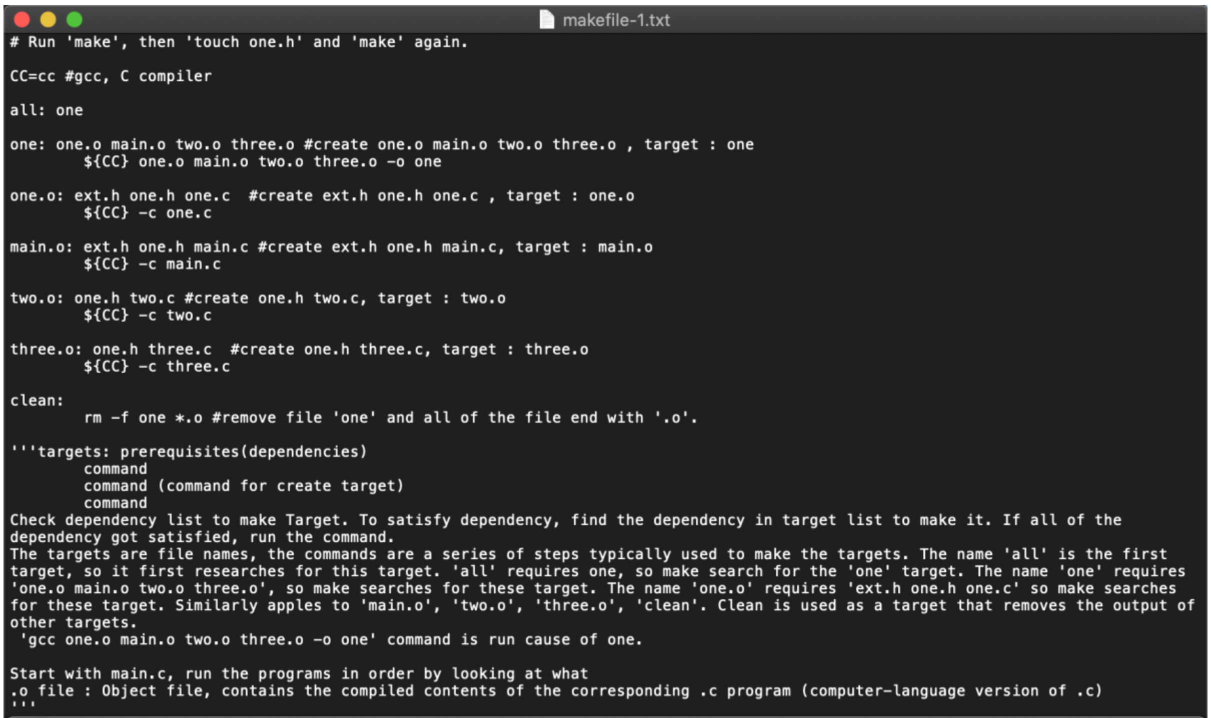
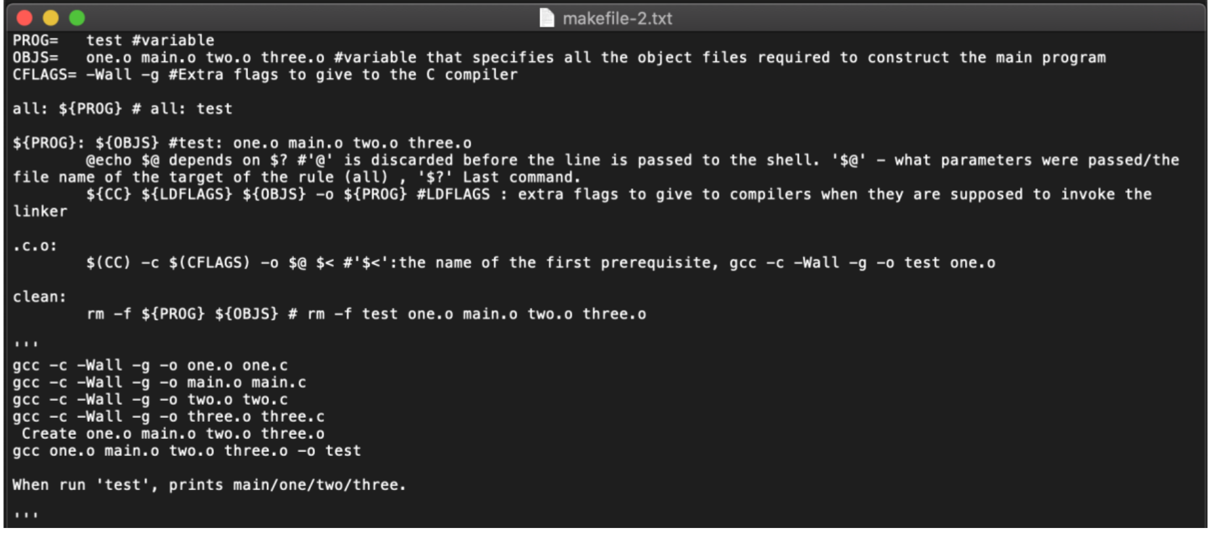
CS 283 : Systems Programming
Lab1

yl3385
Yena Lee

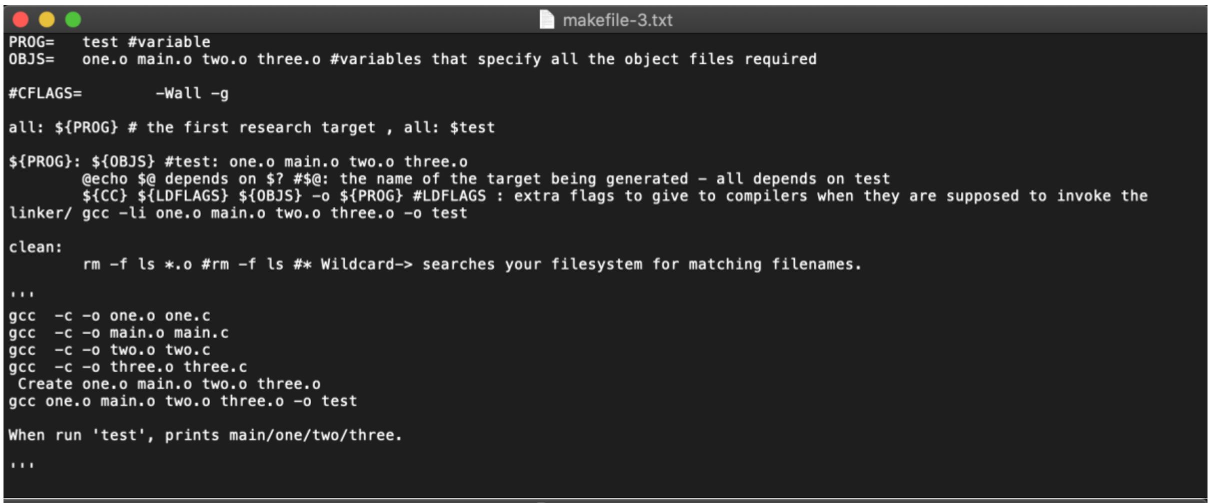
Part 1: Make



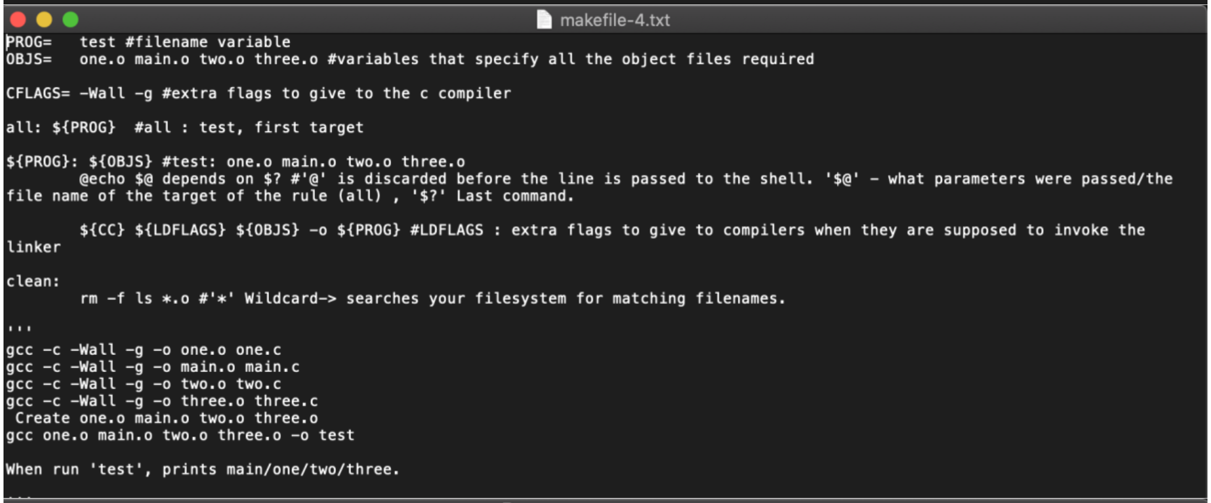
```
# Run 'make', then 'touch one.h' and 'make' again.

CC=cc #gcc, C compiler

all: one

one: one.o main.o two.o three.o #create one.o main.o two.o three.o , target : one
	${CC} one.o main.o two.o three.o -o one

one.o: ext.h one.h one.c  #create ext.h one.h one.c , target : one.o
	${CC} -c one.c

main.o: ext.h one.h main.c #create ext.h one.h main.c, target : main.o
	${CC} -c main.c

two.o: one.h two.c #create one.h two.c, target : two.o
	${CC} -c two.c

three.o: one.h three.c  #create one.h three.c, target : three.o
	${CC} -c three.c

clean:
	rm -f one *.o #remove file 'one' and all of the file end with '.o'.

'''targets: prerequisites(dependencies)
	command
	command (command for create target)
	command
Check dependency list to make Target. To satisfy dependency, find the dependency in target list to make it. If all of the
dependency got satisfied, run the command.
The targets are file names, the commands are a series of steps typically used to make the targets. The name 'all' is the first
target, so it first researches for this target. 'all' requires one, so make search for the 'one' target. The name 'one' requires
'one.o main.o two.o three.o', so make searches for these target. The name 'one.o' requires 'ext.h one.h one.c' so make searches
for these target. Similarly apples to 'main.o', 'two.o', 'three.o', 'clean'. Clean is used as a target that removes the output of
other targets.
 'gcc one.o main.o two.o three.o -o one' command is run cause of one.

Start with main.c, run the programs in order by looking at what
.o file : Object file, contains the compiled contents of the corresponding .c program (computer-language version of .c)
'''
```



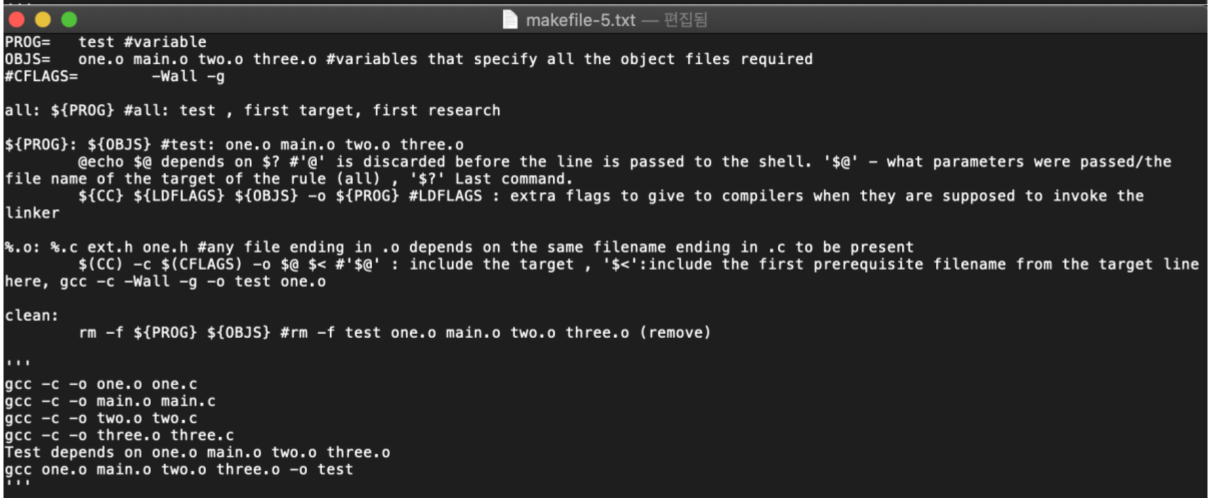
```
PROG=   test #variable
OBJS=   one.o main.o two.o three.o #variable that specifies all the object files required to construct the main program
CFLAGS= -Wall -g #Extra flags to give to the C compiler

all: ${PROG} # all: test

${PROG}: ${OBJS} #test: one.o main.o two.o three.o
	@echo $@ depends on $? #'@' is discarded before the line is passed to the shell. '$@' - what parameters were passed/the
file name of the target of the rule (all) , '$?' Last command.
	${CC} ${LDFLAGS} ${OBJS} -o ${PROG} #LDFLAGS : extra flags to give to compilers when they are supposed to invoke the
linker

.c.o:
	$(CC) -c $(CFLAGS) -o $@ $< #'$<':the name of the first prerequisite, gcc -c -Wall -g -o test one.o

clean:
	rm -f ${PROG} ${OBJS} # rm -f test one.o main.o two.o three.o

'''
gcc -c -Wall -g -o one.o one.c
gcc -c -Wall -g -o main.o main.c
gcc -c -Wall -g -o two.o two.c
gcc -c -Wall -g -o three.o three.c
 Create one.o main.o two.o three.o
gcc one.o main.o two.o three.o -o test

When run 'test', prints main/one/two/three.

'''
```



```
PROG=   test #variable
OBJS=   one.o main.o two.o three.o #variables that specify all the object files required

#CFLAGS=        -Wall -g

all: ${PROG} # the first research target , all: $test

${PROG}: ${OBJS} #test: one.o main.o two.o three.o
        @echo $@ depends on $? #$@: the name of the target being generated - all depends on test
        ${CC} ${LDFLAGS} ${OBJS} -o ${PROG} #LDFLAGS : extra flags to give to compilers when they are supposed to invoke the
linker/ gcc -li one.o main.o two.o three.o -o test

clean:
        rm -f ls *.o #rm -f ls #* Wildcard-> searches your filesystem for matching filenames.

'''
gcc  -c -o one.o one.c
gcc  -c -o main.o main.c
gcc  -c -o two.o two.c
gcc  -c -o three.o three.c
 Create one.o main.o two.o three.o
gcc one.o main.o two.o three.o -o test

When run 'test', prints main/one/two/three.
'''
```



```
PROG=   test #filename variable
OBJS=   one.o main.o two.o three.o #variables that specify all the object files required

CFLAGS= -Wall -g #extra flags to give to the c compiler

all: ${PROG}  #all : test, first target

${PROG}: ${OBJS} #test: one.o main.o two.o three.o
        @echo $@ depends on $? #'@' is discarded before the line is passed to the shell. '$@' - what parameters were passed/the
file name of the target of the rule (all) , '$?' Last command.

        ${CC} ${LDFLAGS} ${OBJS} -o ${PROG} #LDFLAGS : extra flags to give to compilers when they are supposed to invoke the
linker

clean:
        rm -f ls *.o #'*' Wildcard-> searches your filesystem for matching filenames.

'''
gcc -c -Wall -g -o one.o one.c
gcc -c -Wall -g -o main.o main.c
gcc -c -Wall -g -o two.o two.c
gcc -c -Wall -g -o three.o three.c
 Create one.o main.o two.o three.o
gcc one.o main.o two.o three.o -o test

When run 'test', prints main/one/two/three.
```

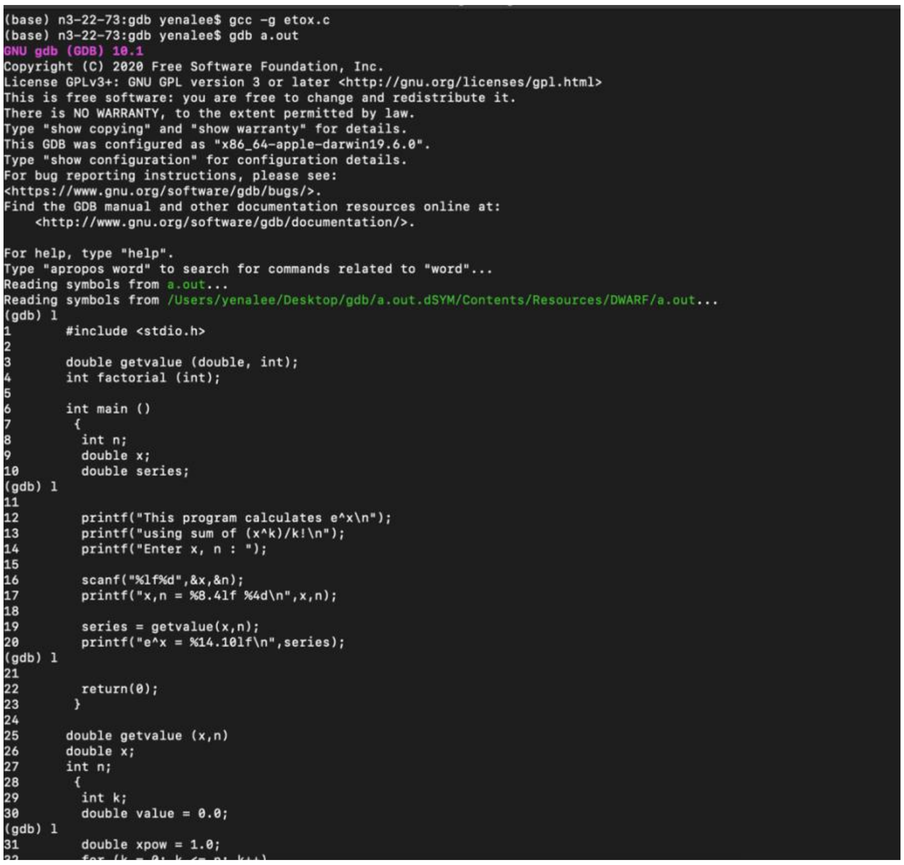


```
PROG=   test #variable
OBJS=   one.o main.o two.o three.o #variables that specify all the object files required
#CFLAGS=        -Wall -g

all: ${PROG} #all: test , first target, first research

${PROG}: ${OBJS} #test: one.o main.o two.o three.o
        @echo $@ depends on $? #'@' is discarded before the line is passed to the shell. '$@' - what parameters were passed/the
file name of the target of the rule (all) , '$?' Last command.
        ${CC} ${LDFLAGS} ${OBJS} -o ${PROG} #LDFLAGS : extra flags to give to compilers when they are supposed to invoke the
linker

%.o: %.c ext.h one.h #any file ending in .o depends on the same filename ending in .c to be present
        $(CC) -c $(CFLAGS) -o $@ $< #'$@' : include the target , '$<':include the first prerequisite filename from the target line
here, gcc -c -Wall -g -o test one.o

clean:
        rm -f ${PROG} ${OBJS} #rm -f test one.o main.o two.o three.o (remove)

'''
gcc -c -o one.o one.c
gcc -c -o main.o main.c
gcc -c -o two.o two.c
gcc -c -o three.o three.c
Test depends on one.o main.o two.o three.o
gcc one.o main.o two.o three.o -o test
'''
```

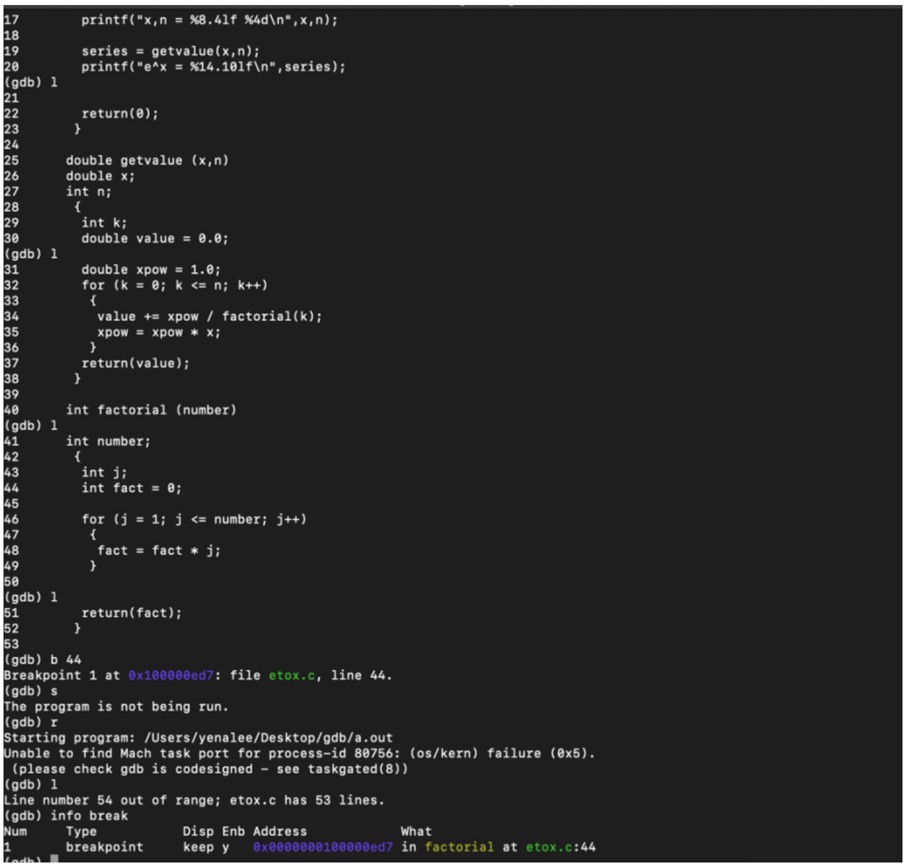
Part 2: GDB

Etox.c

<Interaction with gdb>

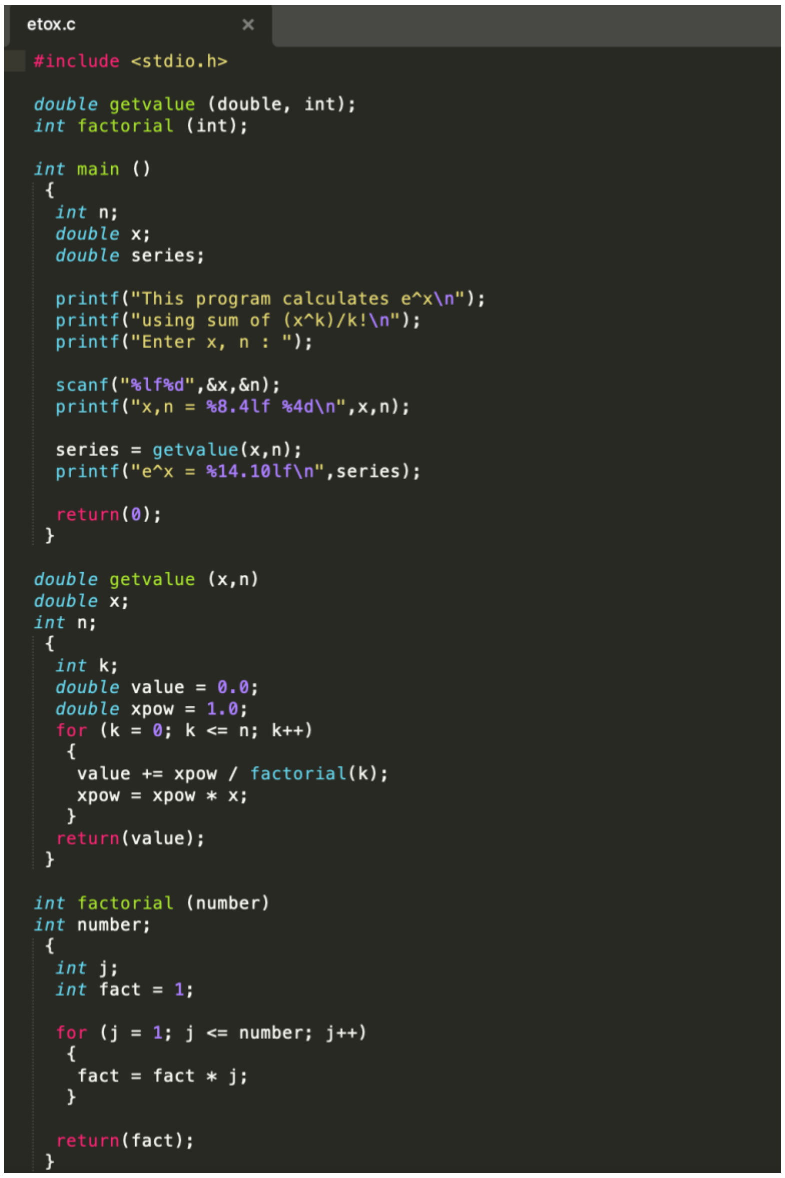


```
(base) n3-22-73:gdb yenalee$ gcc -g etox.c
(base) n3-22-73:gdb yenalee$ gdb a.out
GNU gdb (GDB) 10.1
Copyright (C) 2020 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.
Type "show copying" and "show warranty" for details.
This GDB was configured as "x86_64-apple-darwin19.6.0".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<https://www.gnu.org/software/gdb/bugs/>.
Find the GDB manual and other documentation resources online at:
    <http://www.gnu.org/software/gdb/documentation/>.

For help, type "help".
Type "apropos word" to search for commands related to "word"...
Reading symbols from a.out...
Reading symbols from /Users/yenalee/Desktop/gdb/a.out.dSYM/Contents/Resources/DWARF/a.out...
(gdb) l
1       #include <stdio.h>
2
3       double getvalue (double, int);
4       int factorial (int);
5
6       int main ()
7        {
8         int n;
9         double x;
10        double series;
(gdb) l
11
12        printf("This program calculates e^x\n");
13        printf("using sum of (x^k)/k!\n");
14        printf("Enter x, n : ");
15
16        scanf("%lf%d",&x,&n);
17        printf("x,n = %8.4lf %4d\n",x,n);
18
19        series = getvalue(x,n);
20        printf("e^x = %14.10lf\n",series);
(gdb) l
21
22        return(0);
23       }
24
25      double getvalue (x,n)
26      double x;
27      int n;
28       {
29        int k;
30        double value = 0.0;
(gdb) l
31        double xpow = 1.0;
```



```
17        printf("x,n = %8.4lf %4d\n",x,n);
18
19        series = getvalue(x,n);
20        printf("e^x = %14.10lf\n",series);
(gdb) l
21
22        return(0);
23       }
24
25      double getvalue (x,n)
26      double x;
27      int n;
28       {
29        int k;
30        double value = 0.0;
(gdb) l
31        double xpow = 1.0;
32        for (k = 0; k <= n; k++)
33         {
34          value += xpow / factorial(k);
35          xpow = xpow * x;
36         }
37        return(value);
38       }
39
40      int factorial (number)
(gdb) l
41      int number;
42       {
43        int j;
44        int fact = 0;
45
46        for (j = 1; j <= number; j++)
47         {
48          fact = fact * j;
49         }
50
(gdb) l
51        return(fact);
52       }
53
(gdb) b 44
Breakpoint 1 at 0x100000ed7: file etox.c, line 44.
(gdb) s
The program is not being run.
(gdb) r
Starting program: /Users/yenalee/Desktop/gdb/a.out
Unable to find Mach task port for process-id 80756: (os/kern) failure (0x5).
 (please check gdb is codesigned - see taskgated(8))
(gdb) l
Line number 54 out of range; etox.c has 53 lines.
(gdb) info break
Num     Type           Disp Enb Address            What
1       breakpoint     keep y   0x0000000100000ed7 in factorial at etox.c:44
```

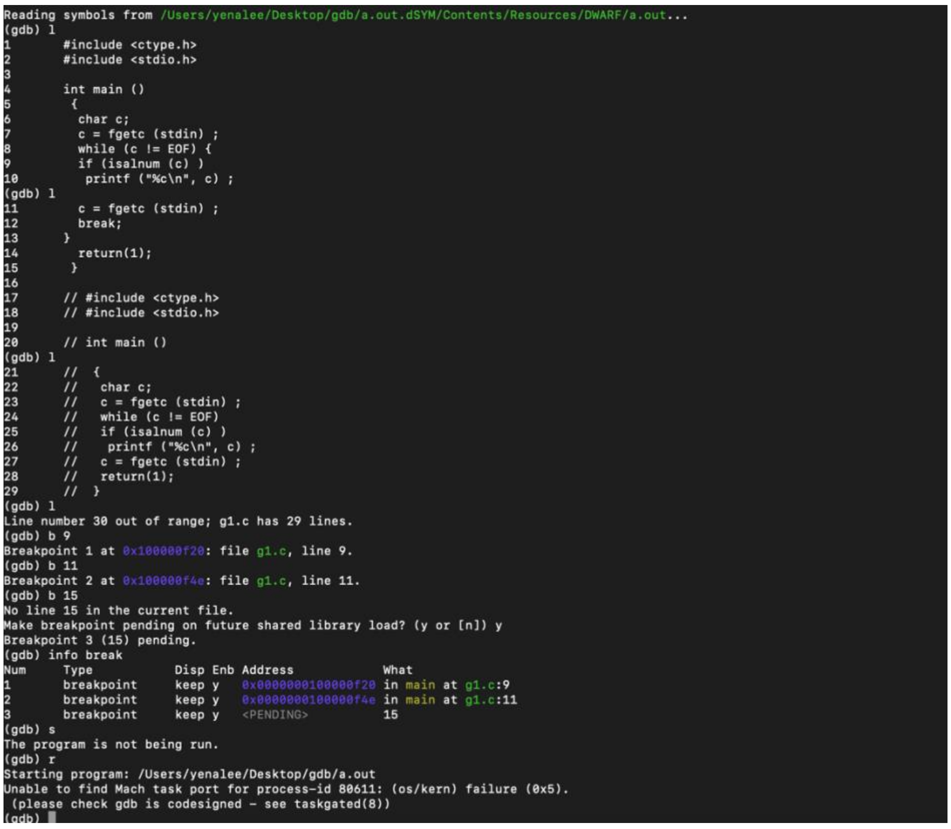
<changed file>


etox.c

```c
#include <stdio.h>

double getvalue (double, int);
int factorial (int);

int main ()
 {
  int n;
  double x;
  double series;

  printf("This program calculates e^x\n");
  printf("using sum of (x^k)/k!\n");
  printf("Enter x, n : ");

  scanf("%lf%d",&x,&n);
  printf("x,n = %8.4lf %4d\n",x,n);

  series = getvalue(x,n);
  printf("e^x = %14.10lf\n",series);

  return(0);
 }

double getvalue (x,n)
double x;
int n;
 {
  int k;
  double value = 0.0;
  double xpow = 1.0;
  for (k = 0; k <= n; k++)
   {
    value += xpow / factorial(k);
    xpow = xpow * x;
   }
  return(value);
 }

int factorial (number)
int number;
 {
  int j;
  int fact = 1;

  for (j = 1; j <= number; j++)
   {
    fact = fact * j;
   }

  return(fact);
 }
```

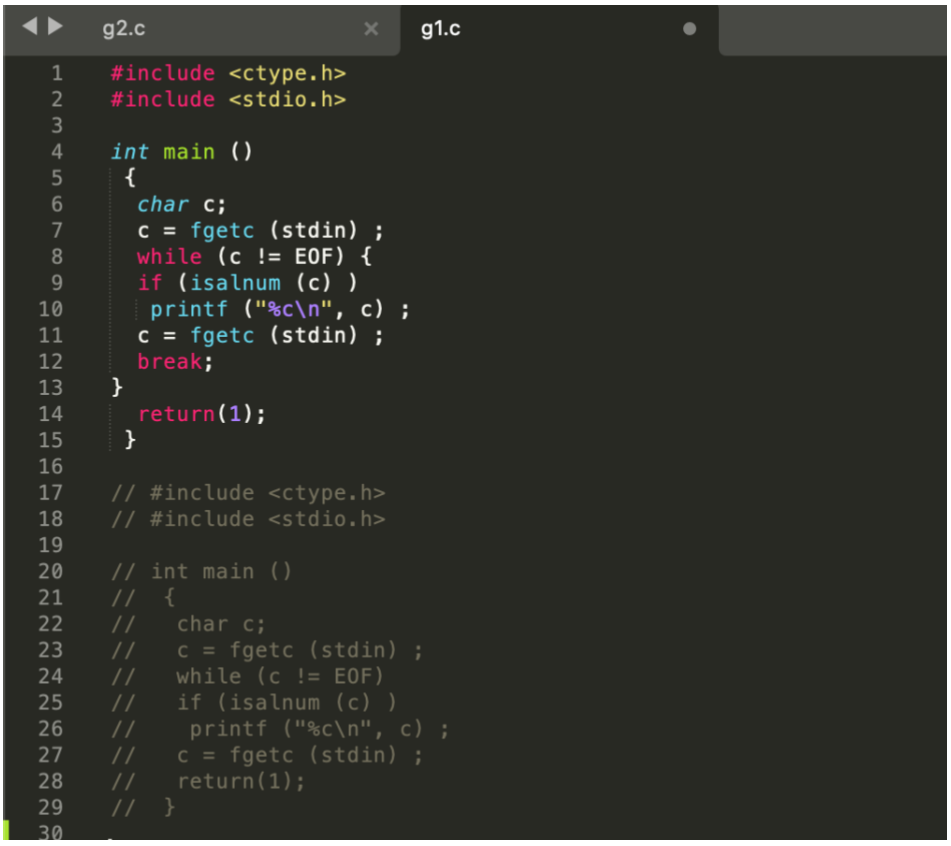
G1.c

<Interaction with gdb>



```
Reading symbols from /Users/yenalee/Desktop/gdb/a.out.dSYM/Contents/Resources/DWARF/a.out...
(gdb) l
1       #include <ctype.h>
2       #include <stdio.h>
3
4       int main ()
5        {
6         char c;
7         c = fgetc (stdin) ;
8         while (c != EOF) {
9         if (isalnum (c) )
10         printf ("%c\n", c) ;
(gdb) l
11        c = fgetc (stdin) ;
12        break;
13      }
14        return(1);
15       }
16
17      // #include <ctype.h>
18      // #include <stdio.h>
19
20      // int main ()
(gdb) l
21      //  {
22      //   char c;
23      //   c = fgetc (stdin) ;
24      //   while (c != EOF)
25      //   if (isalnum (c) )
26      //    printf ("%c\n", c) ;
27      //   c = fgetc (stdin) ;
28      //   return(1);
29      //  }
(gdb) l
Line number 30 out of range; g1.c has 29 lines.
(gdb) b 9
Breakpoint 1 at 0x100000f20: file g1.c, line 9.
(gdb) b 11
Breakpoint 2 at 0x100000f4e: file g1.c, line 11.
(gdb) b 15
No line 15 in the current file.
Make breakpoint pending on future shared library load? (y or [n]) y
Breakpoint 3 (15) pending.
(gdb) info break
Num     Type           Disp Enb Address            What
1       breakpoint     keep y   0x0000000100000f20 in main at g1.c:9
2       breakpoint     keep y   0x0000000100000f4e in main at g1.c:11
3       breakpoint     keep y   <PENDING>          15
(gdb) s
The program is not being run.
(gdb) r
Starting program: /Users/yenalee/Desktop/gdb/a.out
Unable to find Mach task port for process-id 80611: (os/kern) failure (0x5).
 (please check gdb is codesigned - see taskgated(8))
(gdb) 
```

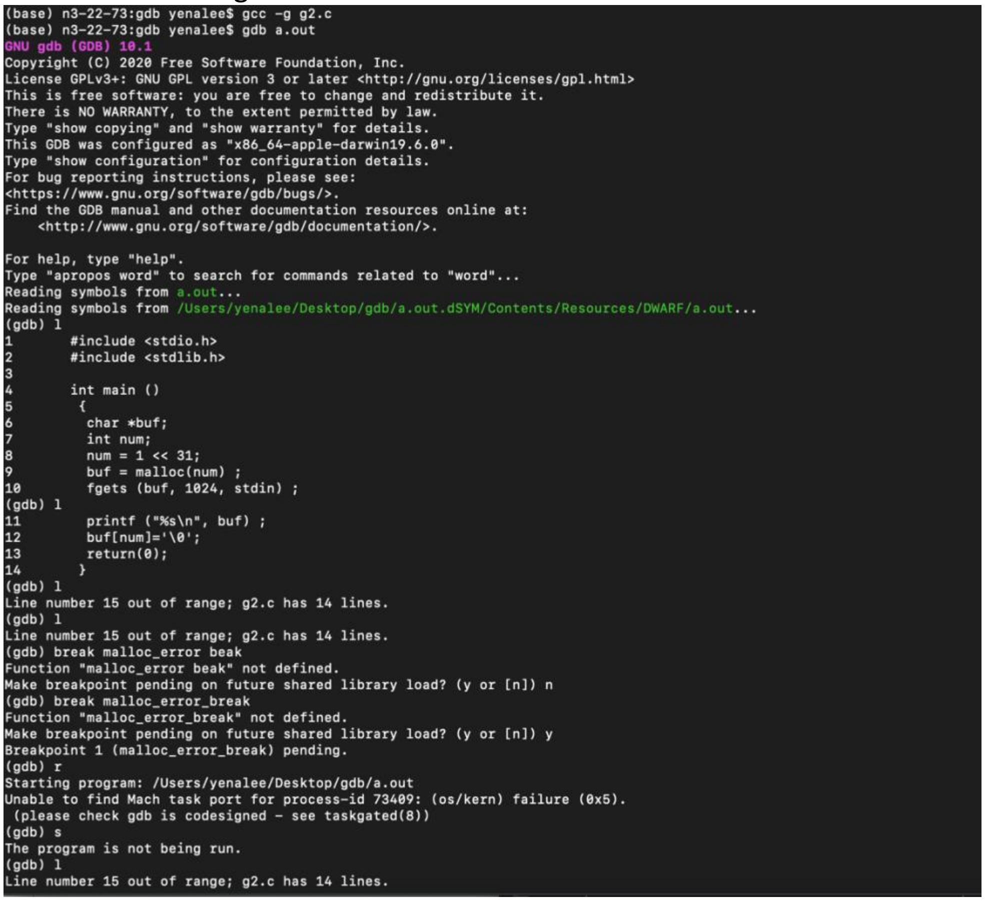
<changed file>



```c
1   #include <ctype.h>
2   #include <stdio.h>
3
4   int main ()
5    {
6     char c;
7     c = fgetc (stdin) ;
8     while (c != EOF) {
9     if (isalnum (c) )
10     printf ("%c\n", c) ;
11    c = fgetc (stdin) ;
12    break;
13  }
14    return(1);
15   }
16
17  // #include <ctype.h>
18  // #include <stdio.h>
19
20  // int main ()
21  //  {
22  //   char c;
23  //   c = fgetc (stdin) ;
24  //   while (c != EOF)
25  //   if (isalnum (c) )
26  //    printf ("%c\n", c) ;
27  //   c = fgetc (stdin) ;
28  //   return(1);
29  //  }
30
```

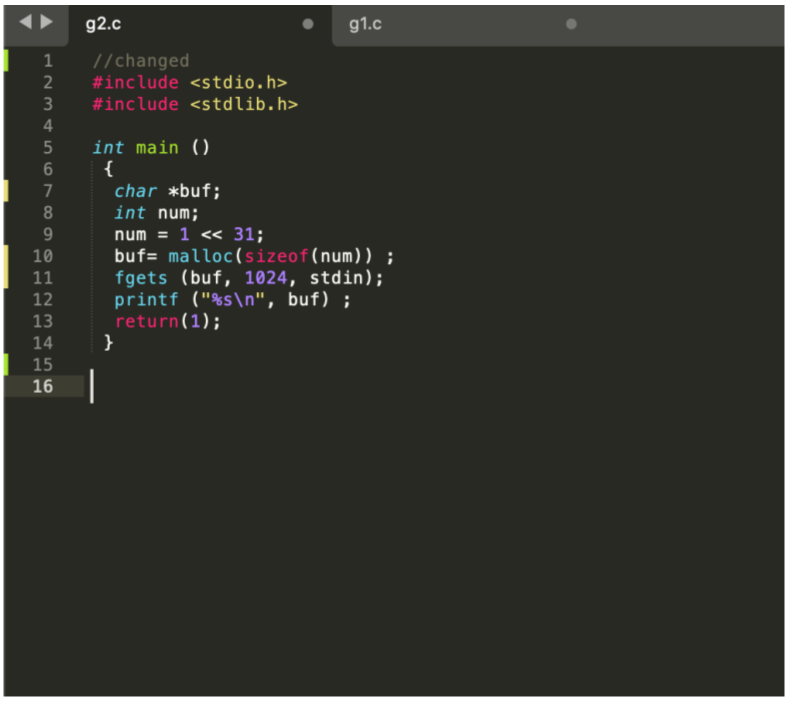
G2.c

<Interaction with gdb>



```
(base) n3-22-73:gdb yenalee$ gcc -g g2.c
(base) n3-22-73:gdb yenalee$ gdb a.out
GNU gdb (GDB) 10.1
Copyright (C) 2020 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.
Type "show copying" and "show warranty" for details.
This GDB was configured as "x86_64-apple-darwin19.6.0".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<https://www.gnu.org/software/gdb/bugs/>.
Find the GDB manual and other documentation resources online at:
    <http://www.gnu.org/software/gdb/documentation/>.

For help, type "help".
Type "apropos word" to search for commands related to "word"...
Reading symbols from a.out...
Reading symbols from /Users/yenalee/Desktop/gdb/a.out.dSYM/Contents/Resources/DWARF/a.out...
(gdb) l
1       #include <stdio.h>
2       #include <stdlib.h>
3
4       int main ()
5        {
6         char *buf;
7         int num;
8         num = 1 << 31;
9         buf = malloc(num) ;
10        fgets (buf, 1024, stdin) ;
(gdb) l
11        printf ("%s\n", buf) ;
12        buf[num]='\0';
13        return(0);
14       }
(gdb) l
Line number 15 out of range; g2.c has 14 lines.
(gdb) l
Line number 15 out of range; g2.c has 14 lines.
(gdb) break malloc_error beak
Function "malloc_error beak" not defined.
Make breakpoint pending on future shared library load? (y or [n]) n
(gdb) break malloc_error_break
Function "malloc_error_break" not defined.
Make breakpoint pending on future shared library load? (y or [n]) y
Breakpoint 1 (malloc_error_break) pending.
(gdb) r
Starting program: /Users/yenalee/Desktop/gdb/a.out
Unable to find Mach task port for process-id 73409: (os/kern) failure (0x5).
 (please check gdb is codesigned - see taskgated(8))
(gdb) s
The program is not being run.
(gdb) l
Line number 15 out of range; g2.c has 14 lines.
```

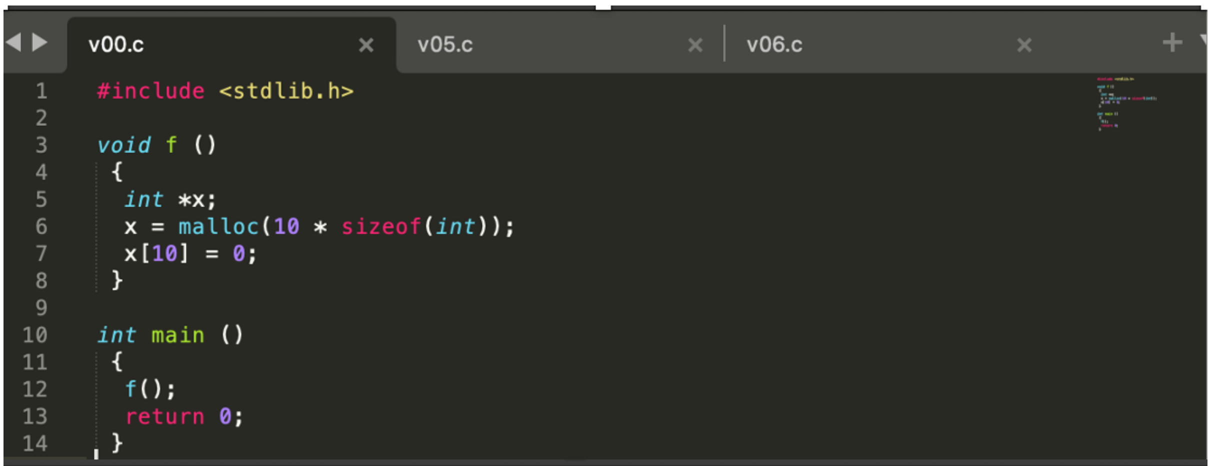
<changed file>



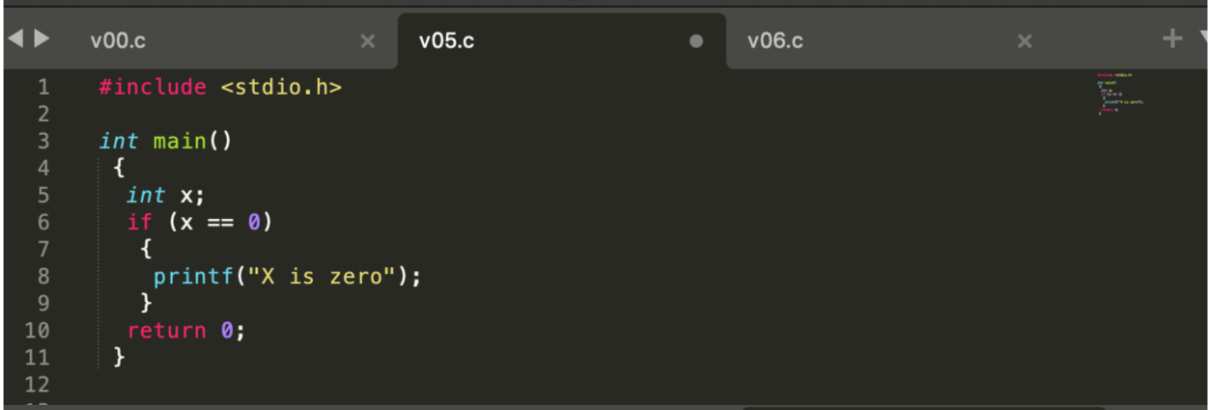
```
//changed
#include <stdio.h>
#include <stdlib.h>

int main ()
 {
  char *buf;
  int num;
  num = 1 << 31;
  buf= malloc(sizeof(num)) ;
  fgets (buf, 1024, stdin);
  printf ("%s\n", buf) ;
  return(1);
 }

```

Part 3: Valgrind

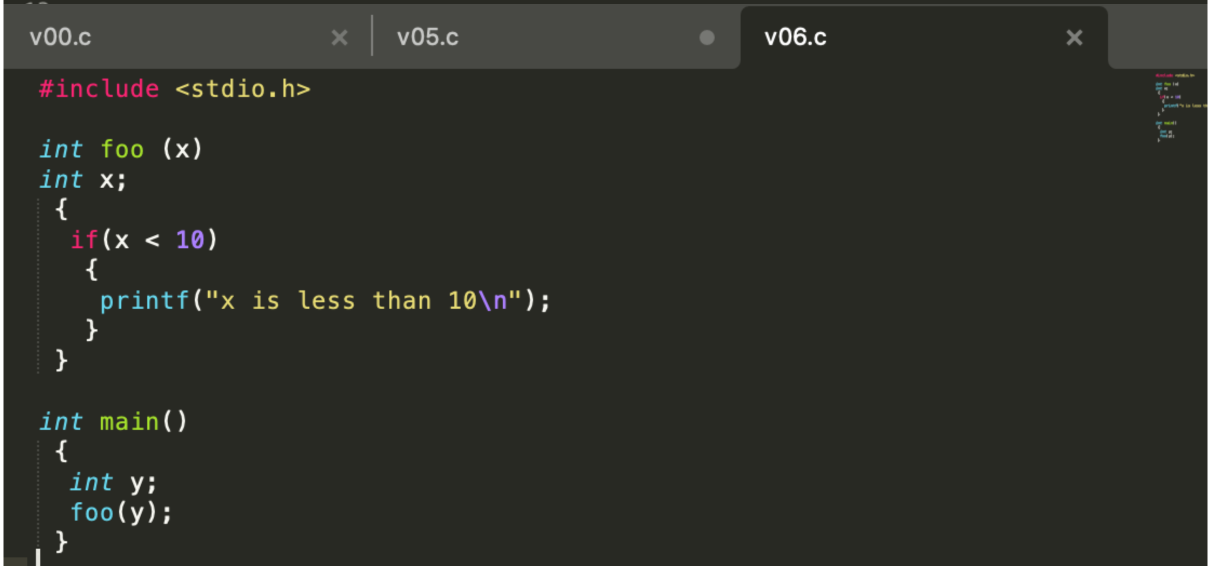
I tried to download Valgrind, but it didn't work. When I googled it, I think it was the problem of mac's version of mine. However, I couldn't figure it out about installing Valgrind, so I couldn't compile the program using valgrind.



```c
#include <stdlib.h>

void f ()
 {
  int *x;
  x = malloc(10 * sizeof(int));
  x[10] = 0;
 }

int main ()
 {
  f();
  return 0;
 }
```



```c
#include <stdio.h>

int main()
 {
  int x;
  if (x == 0)
   {
    printf("X is zero");
   }
  return 0;
 }

```



```c
#include <stdio.h>

int foo (x)
int x;
 {
  if(x < 10)
   {
    printf("x is less than 10\n");
   }
 }

int main()
 {
  int y;
  foo(y);
 }
```