Tutorial 2: Intro To C

Faculty of Engineering and Applied Science

SOFE 3950U : Operating Systems | CRN: 74171

Due: January 29th, 2024

Group 8

Daniel Amasowomwan [100787640]
daniel.amasowomwan@ontariotechu.net

Stanley Watemi [100648403]
stanley.watemi@ontariotechu.net

Fayomi Toyin [100765921]
oluwatoyin.fayomi@ontariotechu.net

Conceptual Questions

1. What are some of the benefits of the C language? Name an example where the C language is used (hint there are many!).
    a. Benefits of the C language:
        i. It’s a universal language
        ii. It works very well with hardware components
        iii. C is fast and efficient
        iv. It can be compiled and run on various platforms etc.
    b. Examples where C is used:
        i. The kernel in Linux, Windows OS, and Mac OS utilizes C
        ii. Most embedded systems are written in C
        iii. Popular databases like MySQL, PostgreSQL, etc. are written in C

2. What is a compiler, what does it do?
    a. A compiler is a program that converts written code in a particular language into machine code that is executable by a computer, bytecode or another programming language.
    b. A compiler performs different stages of analysis and code generation to produce assembly code and passes it to an assembler.

3. What is a makefile, what does it do?
    a. A makefile is a special file containing shell commands. They often contain **rules** telling the system what commands to execute (usually commands to compile a bunch of files). The rules are in two parts. The first line is called a **dependency** line and the next lines are the commands. The commands must be indented with a tab. The dependency line is made of two parts. The first part (before the colon) are target files and the second part (after the colon) are called source files. It is called a dependency line because the first part depends on the second part. Multiple target files must be separated by a space. Multiple source files must also be separated by a space.
    b. After the makefile has been created, a program can be (re)compiled by typing make in the correct directory. Make then reads the makefile and creates a dependency tree and takes whatever action is necessary. It will not necessarily do all the rules in the makefile as all dependencies may not need updated. It will rebuild target files if they are missing or older than the dependency files.

4. Name 5 header files from the C library and explain their purpose.
    a.

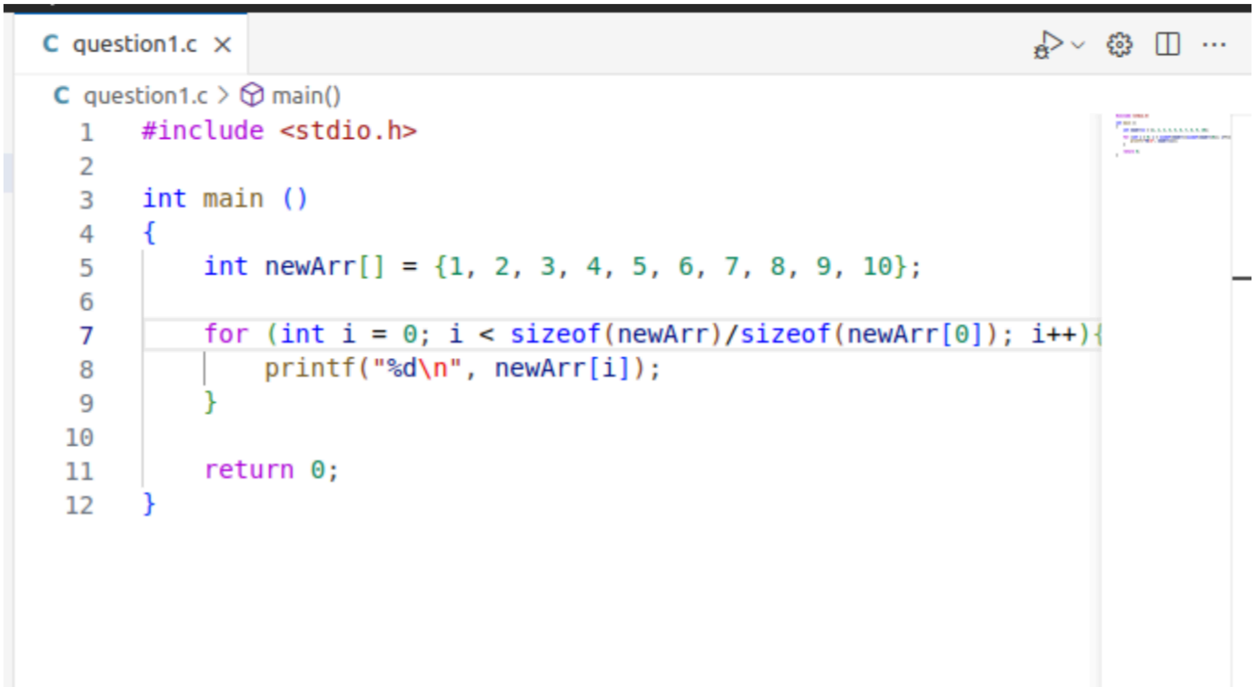
stdio.h	Contains declarations for standard input/output functions
stdlib.h	Contains declarations for memory allocation and management
string.h	Contains declarations for string manipulation functions
math.h	Contains declarations for mathematical functions
time.h	Contains declarations for functions to manipulate date/time

5. Lookup one function from each of the header files and put down the function name and describe what it does
    a.

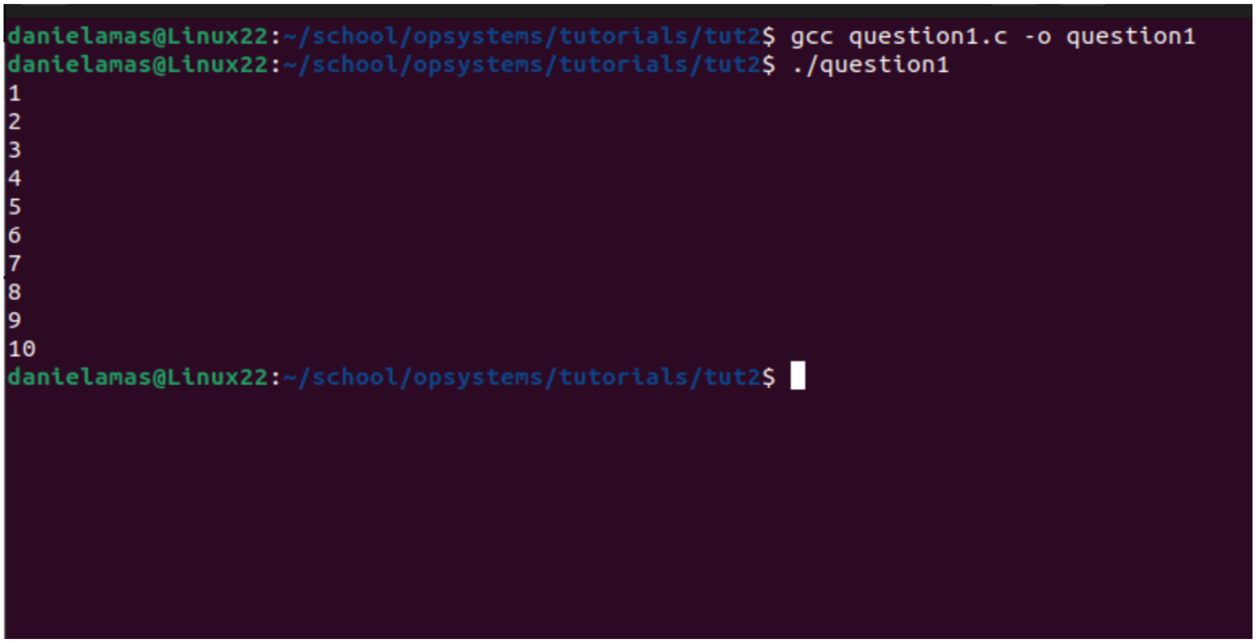
stdio.h	int fclose(FILE *stream)	Closes the stream. All buffers are flushed.
stdlib.h	void *calloc(size_t nitems, size_t size)	Allocates the requested memory and returns a pointer to it.
string.h	void *memchr(const void *str, int c, size_t n)	Searches for the first occurrence of the character c (an unsigned char) in the first n bytes of the string pointed to, by the argument str.
math.h	double pow(double x, double y)	Returns x raised to the power of y.
time.h	clock_t clock(void)	Returns the processor clock time used since the beginning of an implementation defined era (normally the beginning of the program).

Application Questions

1. Create a program that does the following
    - Creates an array of 10 integer values from 1 to 10
    - Has a loop that iterates through the array and prints each integer in the array

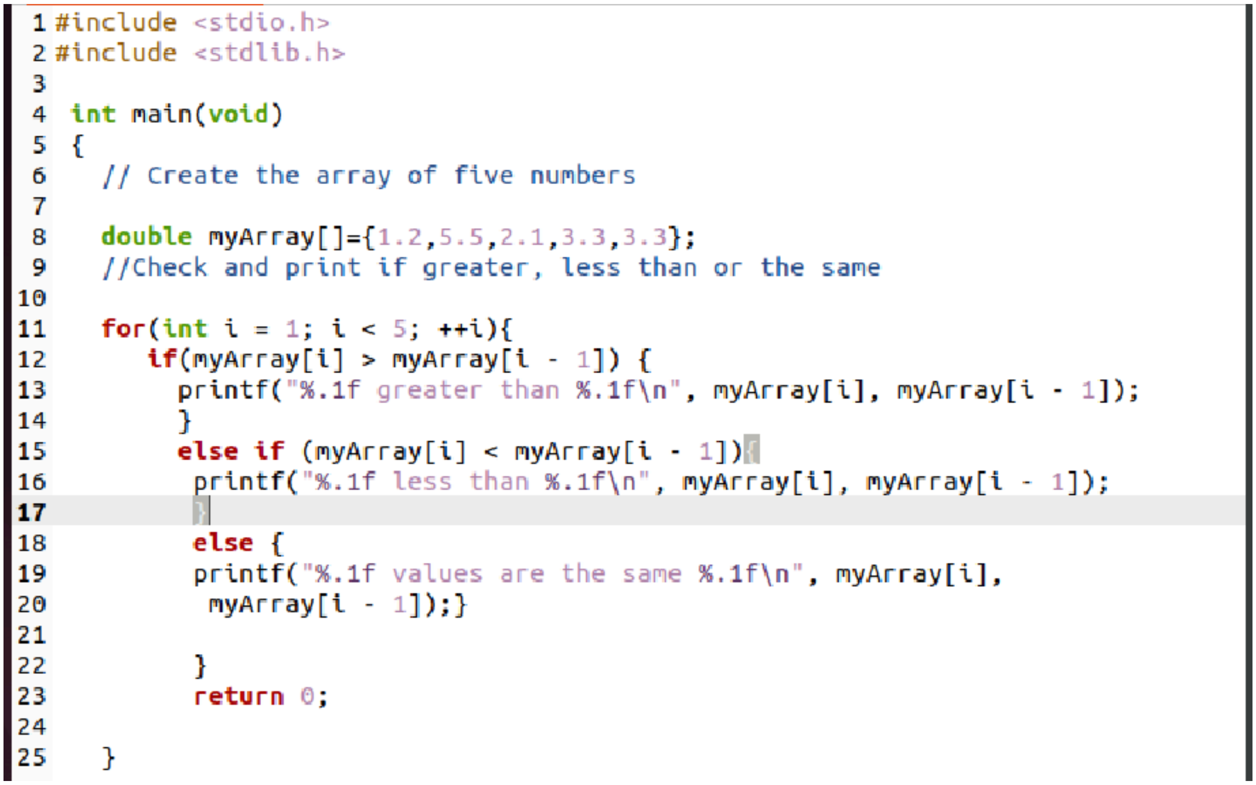


```c
#include <stdio.h>

int main ()
{
    int newArr[] = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10};

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

    return 0;
}
```

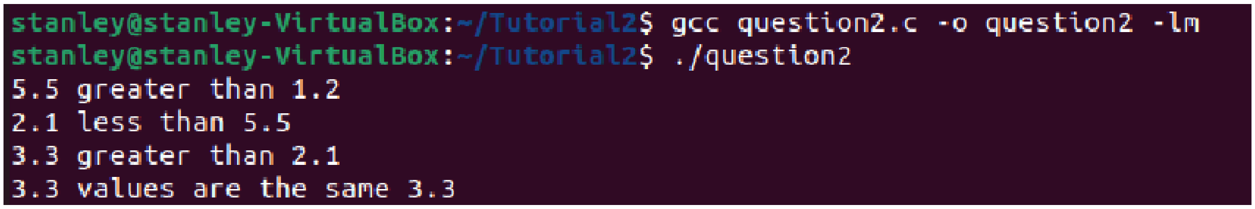


```
danielamas@Linux22:~/school/opsystems/tutorials/tut2$ gcc question1.c -o question1
danielamas@Linux22:~/school/opsystems/tutorials/tut2$ ./question1
1
2
3
4
5
6
7
8
9
10
danielamas@Linux22:~/school/opsystems/tutorials/tut2$ 
```

2. Create a program that does the following
   - Creates an array of the following 5 double values 1.2, 5.5, 2.1, 3.3, 3.3
   - Checks if the current value is greater than, less than, or the same as the previous value and for each case prints either "greater than", "less than", or "the same"

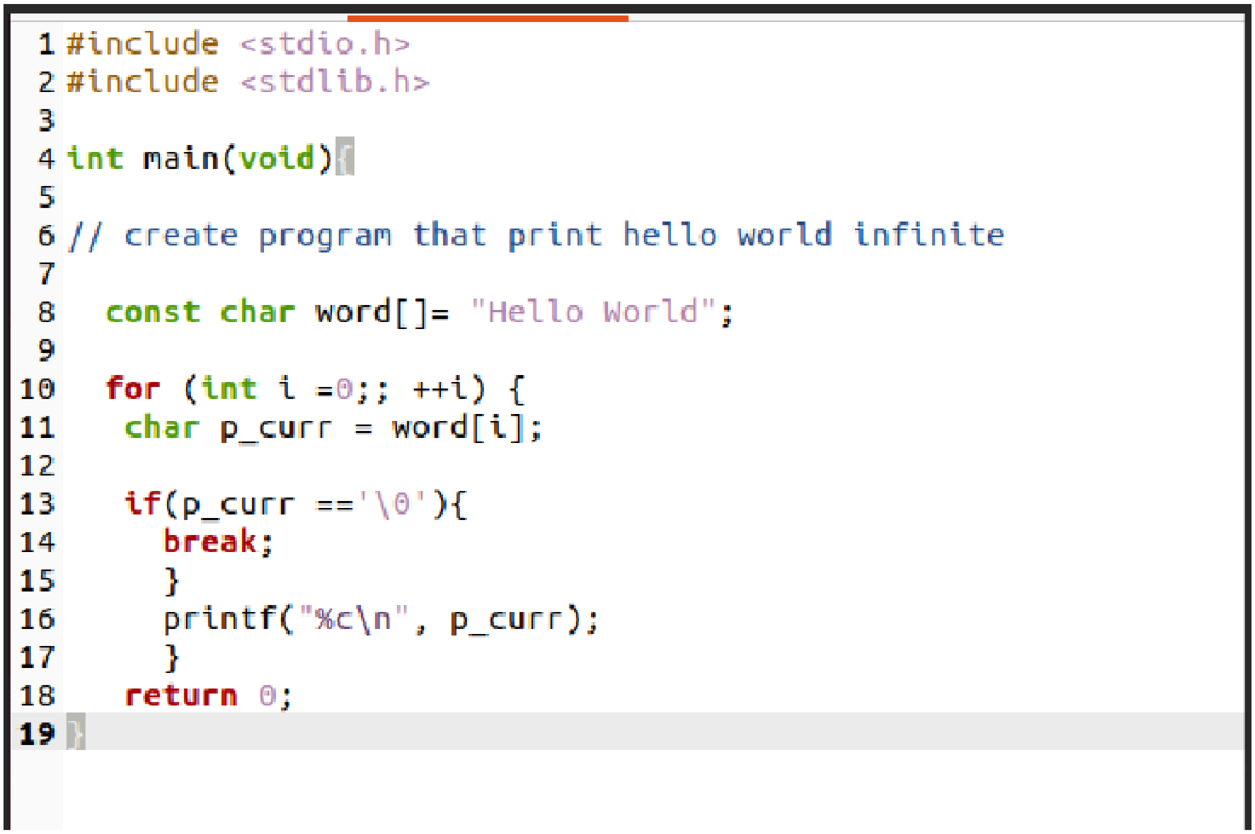


```
#include <stdio.h>
#include <stdlib.h>

 int main(void)
 {
   // Create the array of five numbers

   double myArray[]={1.2,5.5,2.1,3.3,3.3};
   //Check and print if greater, less than or the same

   for(int i = 1; i < 5; ++i){
      if(myArray[i] > myArray[i - 1]) {
        printf("%.1f greater than %.1f\n", myArray[i], myArray[i - 1]);
        }
        else if (myArray[i] < myArray[i - 1]){
         printf("%.1f less than %.1f\n", myArray[i], myArray[i - 1]);
         }
         else {
         printf("%.1f values are the same %.1f\n", myArray[i],
          myArray[i - 1]);}

         }
         return 0;

   }
```

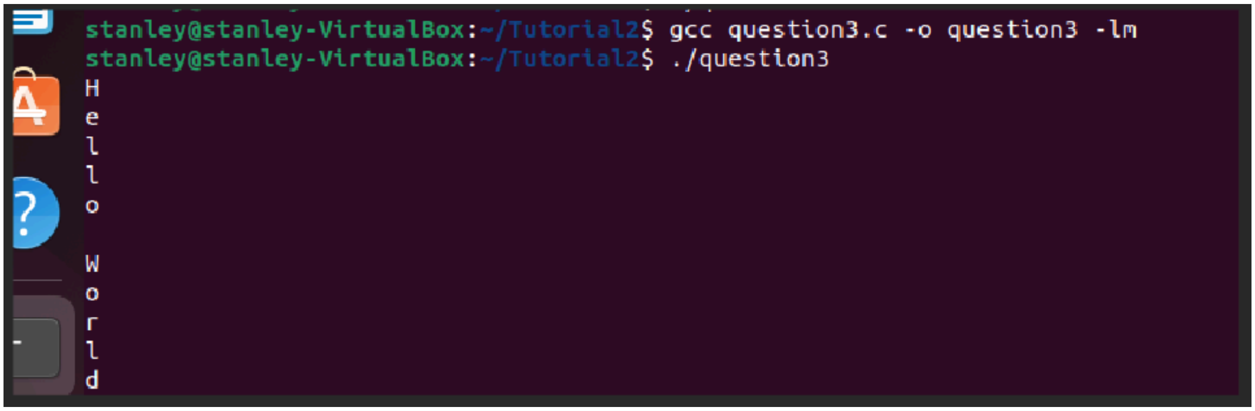


```
stanley@stanley-VirtualBox:~/Tutorial2$ gcc question2.c -o question2 -lm
stanley@stanley-VirtualBox:~/Tutorial2$ ./question2
5.5 greater than 1.2
2.1 less than 5.5
3.3 greater than 2.1
3.3 values are the same 3.3
```

3. Create a program that does the following
   - Has a string (character array) containing the words "hello world"
   - Performs an infinite loop that prints each character in the string until it reaches the end of the string (hint strings are terminated by something...)
   - Your resultant output should print the string "hello world"

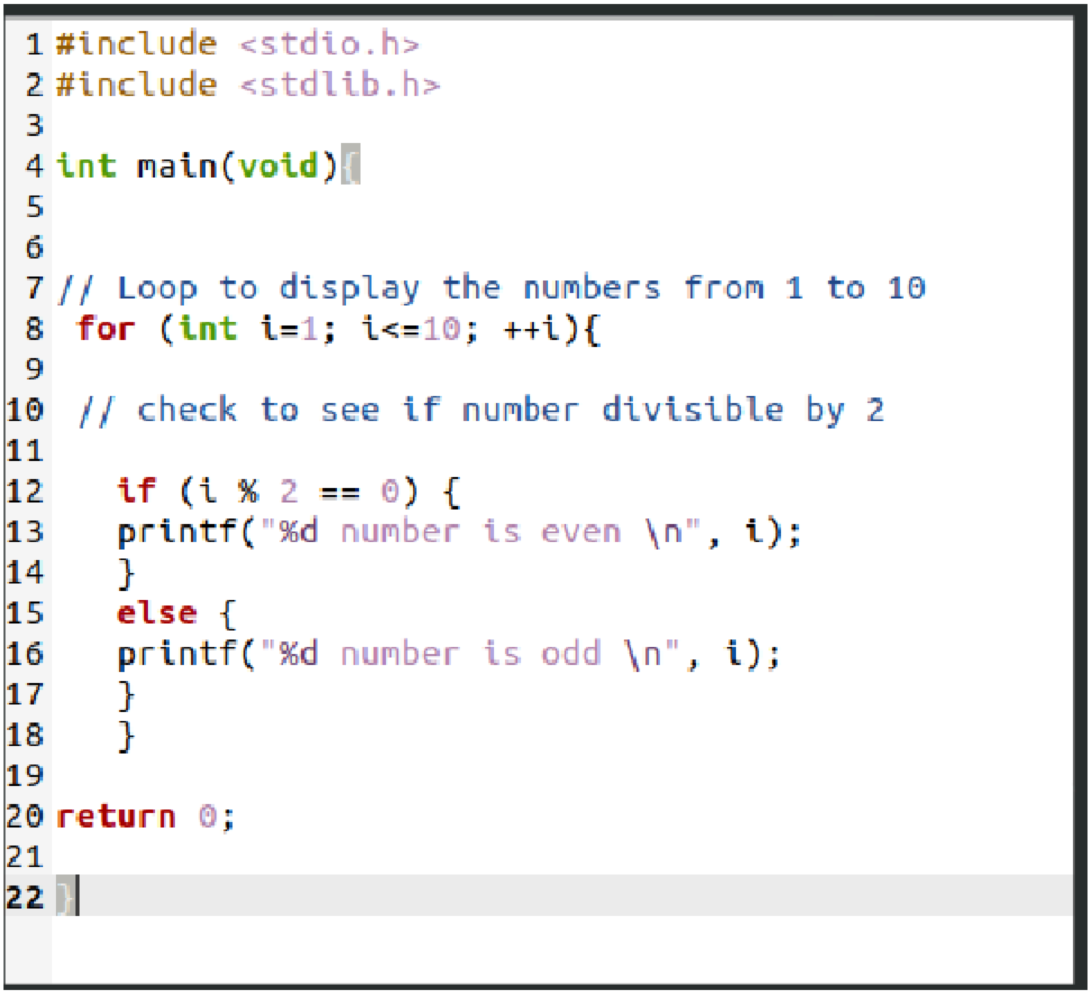


```c
#include <stdio.h>
#include <stdlib.h>

int main(void){

// create program that print hello world infinite

  const char word[]= "Hello World";

  for (int i =0;; ++i) {
   char p_curr = word[i];

   if(p_curr =='\0'){
     break;
     }
     printf("%c\n", p_curr);
     }
   return 0;
}
```

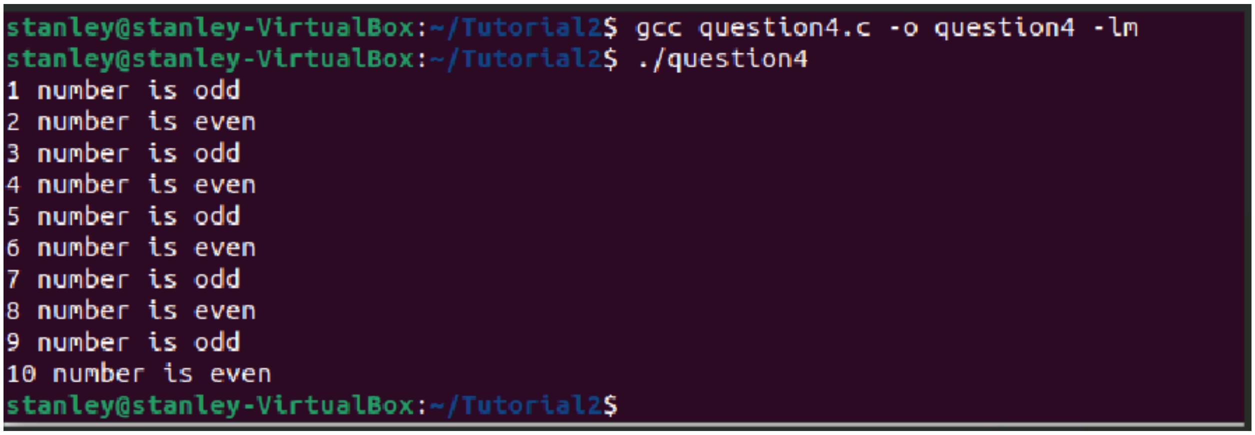


```
stanley@stanley-VirtualBox:~/Tutorial2$ gcc question3.c -o question3 -lm
stanley@stanley-VirtualBox:~/Tutorial2$ ./question3
H
e
l
l
o
 
W
o
r
l
d
```

4. Create a program that does the following
    - Loops from 1 to 10 (NOT 0 - 9)
    - Checks if the number is even or odd
    - Prints the number and the word "even" or "odd" based on whether the number is even or odd



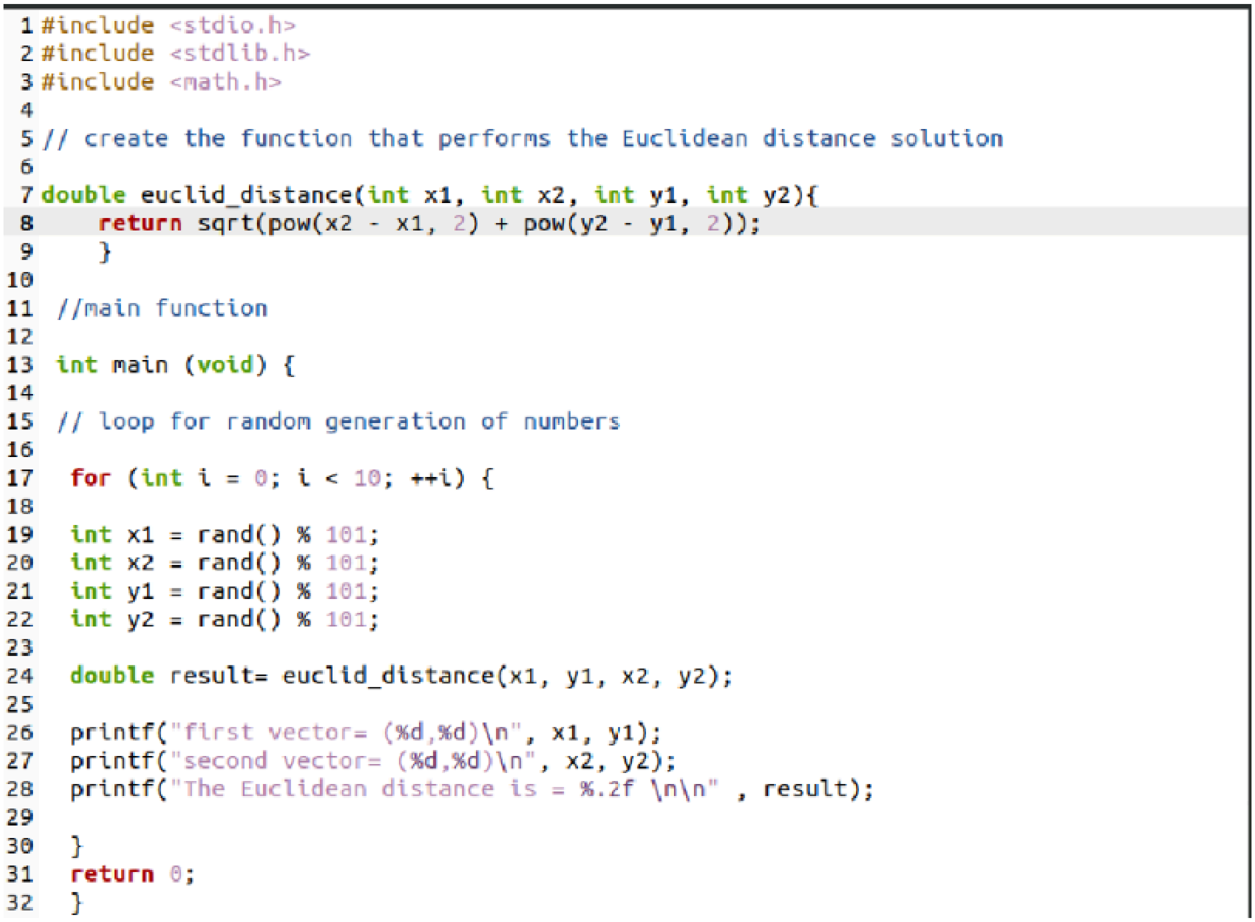
```c
#include <stdio.h>
#include <stdlib.h>

int main(void){


// Loop to display the numbers from 1 to 10
 for (int i=1; i<=10; ++i){

 // check to see if number divisible by 2

   if (i % 2 == 0) {
   printf("%d number is even \n", i);
   }
   else {
   printf("%d number is odd \n", i);
   }
   }

return 0;

}
```



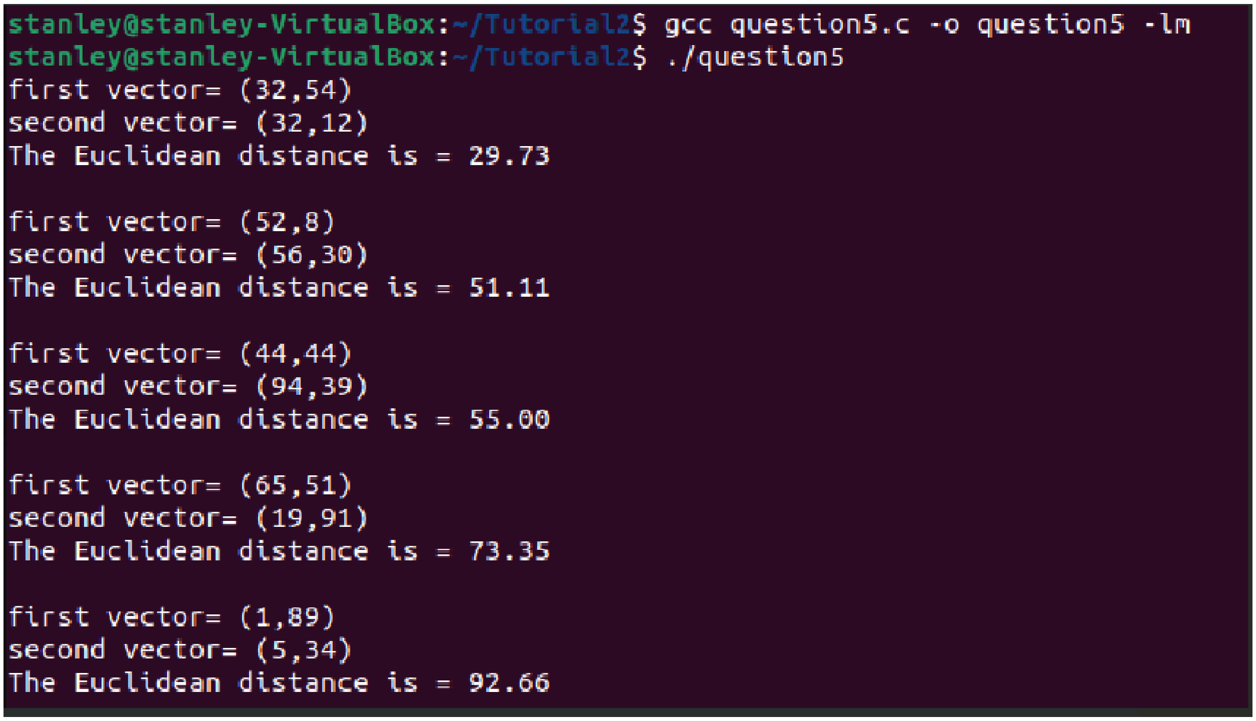
```
stanley@stanley-VirtualBox:~/Tutorial2$ gcc question4.c -o question4 -lm
stanley@stanley-VirtualBox:~/Tutorial2$ ./question4
1 number is odd
2 number is even
3 number is odd
4 number is even
5 number is odd
6 number is even
7 number is odd
8 number is even
9 number is odd
10 number is even
stanley@stanley-VirtualBox:~/Tutorial2$
```

5. A bit of math, create a program that does the following

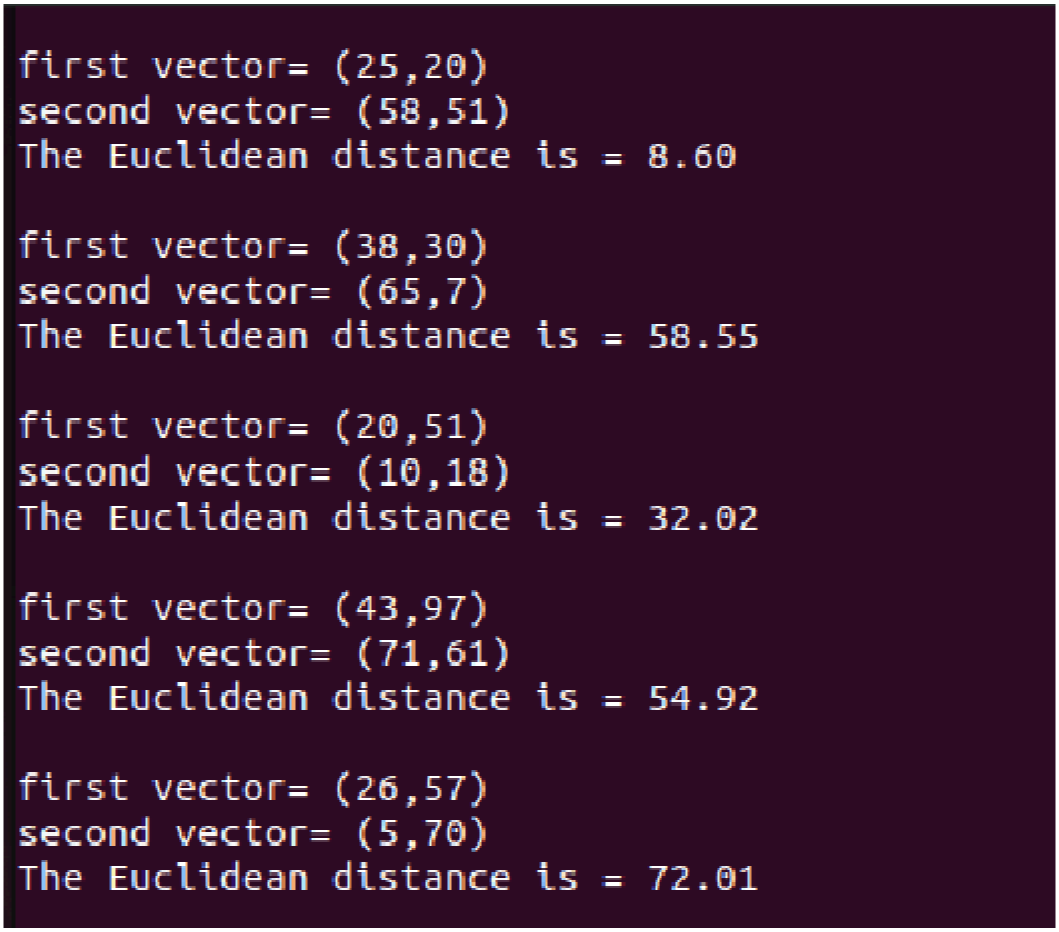
- Has a function called euclid_dist that takes two sets of vectors containing x and y that are integers and returns a double.
    - euclid_dist(x1, y1, x2, y2)
    - The function computes the euclidean distance between the two vectors containing x and y coordinates, you will need to lookup the documentation in math.h to calculate the euclidean distance.
    - The function returns the result of the calculation
- The program then has a loop which does the following
    - Loops 10 times
    - Each time generates a random number between 0 and 100 for each x and y variable for the two vectors.
    - Calls the euclid_dist function and gets the results
    - Prints the two vectors and the euclidean distance to the console
    - Hint you will need to use the C reference site: http://en.cppreference.com/w/c and look up the math.h and stdlib.h libraries for the mathematical functions and the rand() function, add #include <math.h> and #include <stdlib.h> to your file to use these functions.
- In order to compile your program you will need to add the argument -lm which tells the compiler to link to the math library.



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

// create the function that performs the Euclidean distance solution

double euclid_distance(int x1, int x2, int y1, int y2){
    return sqrt(pow(x2 - x1, 2) + pow(y2 - y1, 2));
    }

//main function

int main (void) {

// loop for random generation of numbers

 for (int i = 0; i < 10; ++i) {

 int x1 = rand() % 101;
 int x2 = rand() % 101;
 int y1 = rand() % 101;
 int y2 = rand() % 101;

 double result= euclid_distance(x1, y1, x2, y2);

 printf("first vector= (%d,%d)\n", x1, y1);
 printf("second vector= (%d,%d)\n", x2, y2);
 printf("The Euclidean distance is = %.2f \n\n" , result);

 }
 return 0;
 }
```



```
stanley@stanley-VirtualBox:~/Tutorial2$ gcc question5.c -o question5 -lm
stanley@stanley-VirtualBox:~/Tutorial2$ ./question5
first vector= (32,54)
second vector= (32,12)
The Euclidean distance is = 29.73

first vector= (52,8)
second vector= (56,30)
The Euclidean distance is = 51.11

first vector= (44,44)
second vector= (94,39)
The Euclidean distance is = 55.00

first vector= (65,51)
second vector= (19,91)
The Euclidean distance is = 73.35

first vector= (1,89)
second vector= (5,34)
The Euclidean distance is = 92.66
```



```
first vector= (25,20)
second vector= (58,51)
The Euclidean distance is = 8.60

first vector= (38,30)
second vector= (65,7)
The Euclidean distance is = 58.55

first vector= (20,51)
second vector= (10,18)
The Euclidean distance is = 32.02

first vector= (43,97)
second vector= (71,61)
The Euclidean distance is = 54.92

first vector= (26,57)
second vector= (5,70)
The Euclidean distance is = 72.01
```