```
/*
* This program returns the largest and second largest number from a list of 10
inputs provided by the user.
* Written by Virginia Gonzalez
* September 1, 2022
*/
#include <stdio.h>
int long long Fibonacci(int nthInt);
int main( void )
{
   int long long nth = 0;
   printf("Enter an nth integer: ");
   scanf("%lld", &nth);

   printf("%lld", Fibonacci(nth));
   printf("\n");

    return 0; //we return 0 if the program ends succesfully
} //end function main

int long long Fibonacci (int index)
{
      if (index == 1)
            return 0;
      else if (index == 2)
            return 1;
      else
            return Fibonacci(index-1) + Fibonacci(index-2);

}
```

```
/*
* This program takes two 3x3 matrices to return their addition and subtraction.
* It also takes the transpose of any 3x3 matrix
*
#include<stdio.h>

void matrixAdd(int a[3][3], int b[3][3], int sum[3][3]);
void matrixSubtract(int a[3][3], int b[3][3], int diff[3][3]);
void matrixTranspose(int a[3][3], int trans[3][3]);
//void matrixMultiply(int a[3][3], int b[3][3], int prod[3][3]);

int main(void)
{

   int a[3][3] = { {1,2,3}, {4,5,6}, {7,8,9} };
   int b[3][3] = { {3,5,7}, {7,3,2}, {0,2,6} };
   int sum[3][3];
   int diff[3][3];
   int trans[3][3];
   int prod[3][3];

   matrixAdd(a, b, sum);
   matrixSubtract(a, b, diff);
   matrixTranspose(a, trans);
  // matrixMultiply(a, b, prod);

}

void matrixAdd(int a[3][3], int b[3][3], int sum[3][3])
{
   printf("Matrix addition: \n");

   for (int i = 0; i < 3 ; i++)
   {
      for (int j = 0; j < 3 ; j++)
      {
         sum[i][j] = a[i][j] + b[i][j];
         printf("%d ", sum[i][j]);
      }

       printf("\n");

   }

   printf("\n\n");
}
```

```c
#include <stdio.h>

void swap(int *num1Ptr, int *num2Ptr);
int main(void)
{

   int num1, num2;

   printf("Please enter an integer \n");
   scanf("%d", &num1);
   printf("Please enter a second integer \n");
   scanf("%d", &num2);

   swap(&num1, &num2);

   printf("\nAfter swapping: \n");
   printf("The first integer is %d \n", num1);
   printf("The second integer is %d \n", num2);
}

void swap(int *num1Ptr, int *num2Ptr)
{
   int temp = *num1Ptr;
   *num1Ptr = *num2Ptr;
   *num2Ptr = temp;
}
```

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

struct Student
{
   char name[15];
   int age;
   float gpa;
};

int main(void)
{
   struct Student structArray[5];

  // studentsPtr = malloc(sizeof(struct Student));

   for (int i=0 ; i<5 ; i++)
   {
     printf("Please enter student's first name: ");
     scanf("%s", structArray[i].name);
     printf("Enter %s's age: ", structArray[i].name);
     scanf("%d", &structArray[i].age);
     printf("Enter %s's GPA: ", structArray[i].name);
     scanf("%f", &structArray[i].gpa);
   }

   printf("\n");

   for(int i=1; i <+ 5; i++)
   {
      printf("Student Record #%d: \n", i);
      printf("%s\n", structArray[i].name);
      printf("%d\n", structArray[i].age);
      printf("%.2f\n\n", structArray[i].gpa);
   }


}
```

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

// Function prototypes
int RollDice();
void UserStatus(int sum);

int main(void)
{
   // User input variables
   srand(time(NULL));
   char ans;
   int r1, r2;

   // Display program concept
   printf("\nThis program mimics the game of Craps\n");
   printf("Craps is a dice game in which players bet on\n");
   printf("the outcomes of the roll of a pair of dice\n");
   printf("Press Y or y to play: ");
   scanf("%c", &ans); // Save input from the user

   // If user wants to play, then proceed
   if (ans == 'Y' || ans == 'y')
   {
      // Obtain two random integer variables
      r1 = RollDice();
      r2 = RollDice();

      // Print the output of the random variables and its sum
      printf("\nRoll#1\n");
      printf("First Die: %d\n", r1);
      printf("Second Die: %d\n", r2);
      printf("Sum: %d\n",r1+r2);

      // Call the function to check if the user won or lost
      UserStatus(r1+r2);
   }
   // If the user doesn't want to play, then exit the program
    else
       return (0);

}
//  Function to return a random integer
int RollDice()
{
   int r = rand() % 6 + 1;
   return r;
}
```

```
// Function to display if the user won or lost
void UserStatus(int sum)
{
   // Seed the random number generator
   srand(time(NULL));

   // Variables
   int firstPt = sum;
   int r1, r2, secondPt;
   int counter = 2;

   // enum data type called status where
   // WIN=1, LOSE=2, and CONTINUE=3
   enum status{WIN, LOSE, CONTINUE};
   enum status outcome;

   // If the sum of first roll is either 7 or 11
   if ( sum==7 || sum==11 )
      outcome = WIN;
   // If the sum of the first roll is either 2,3, or 12
   else if ( sum==2 || sum==3 || sum==12 )
      outcome = LOSE;
   // All other possibilities
   else
      outcome = CONTINUE;

   // Display the corresponding status of outcome
   switch(outcome)
   {
      case WIN:
         printf("Congratulations! You won!\n");
         break;

      case LOSE:
         printf("Sorry, you lost!\n");
         break;

      case CONTINUE:
         // Continuous loop until the sum of the first roll or 7 is reached
         do {
            printf("\n-----------------\n");
            r1 = RollDice();
            r2 = RollDice();
            printf("Roll#%d\n", counter++);
            printf("First Die: %d\n", r1);
            printf("Second Die: %d\n", r2);
            printf("Sum: %d\n",r1+r2);
            secondPt = r1+r2;
         }while( !(secondPt==7) && !(secondPt==firstPt) );
```

```
         // Display the outcome of the last roll
         if (secondPt == 7)
            printf("Sorry, you lost!\n");
         else
            printf("Congratulations! You won!\n");
         break;

   }

}

/*
* This program is designed to gather two inputs from the user and
* return the difference of the two dates. Struct date is used to
* save the user input. The function daysElapsed takes a struct date
* data type to calculate the number of days for a given date and
* then returns that number back to main.
*
* Created by Virginia Gonzalez
* October 10, 2022
*/
#include <stdio.h>
#include <stdlib.h>

struct date
{
    int month, date, year;
} aDate, bDate;

int daysElapsed(struct date xDate)
{
  // variable to hold number of days
  int nDays;

  // to determine which year to use
  if (xDate.month <= 2)
    xDate.year = xDate.year - 1;

  // to determine which month to use
  if (xDate.month <= 2)
    xDate.month = xDate.month + 13;
  else
    xDate.month = xDate.month + 1;

  // formula to return the number of days for a given date
  nDays = 1461 * xDate.year/4 + 153 * xDate.month/5 + xDate.date;

  return nDays;
}
```

```
int main(void)
{
  // variables to hold number of days
  int date1Days, date2Days, daysBetween;

  // initializing structs
  struct date aDate = {0,0,0};
  struct date bDate = {0,0,0};

  // prompt the user to input two dates
  printf("\nThis program provides the number of days between two dates. \n");
  printf("------------------------------------------------------------ \n\n");
  printf("Please enter the first date (mm/dd/yyyy): ");
  scanf("%d/%d/%d", &aDate.month,&aDate.date,&aDate.year);
  printf("Please enter the second date (mm/dd/yyyy): ");
  scanf("%d/%d/%d", &bDate.month,&bDate.date,&bDate.year);

  // calculates days for date1 and date2 and returns the difference
  date1Days = daysElapsed(aDate);
  date2Days = daysElapsed(bDate);
  daysBetween = abs(date1Days-date2Days);

  // output answer
  printf("\nThere are %d between the two dates.", daysBetween);

  return 0;
}
```

```
import sys


# this function converts C to F and places the new values to a Fahrenheit list
def convert(list_c):
    list_f=[]
    for x in list_c:
        y = x * (9.0 / 5.0) + 32
        list_f.append(y)
    return list_f


# the 'main' point at which code execution will start:
def main():
    c_min = int(input('Enter the minimum value: '))
    c_max = int(input('Enter the maximum value: '))

    c_range = range(c_min, c_max+1, 1)
    list_c = list(c_range)
    list_f = convert(list_c)

    print("First number: " + str(c_min))
    print("Second number: " + str(c_max))
    print(" C       F")

    # key-value pair of C and F and prints as two columns
    for (x,y) in zip(list_c,list_f):
        print(x,'   ',y)

# The following statements indicate that the starting point of our execution is
main.
# We will discuss them in more detail over the next week.


if __name__ == "__main__":
    sys.exit(main())
```

```
# Created a Monster class which contains overloaded methods
class Monster:
    def __init__(self, monsterName, monsterSize, monsterLocation):
        self.name = monsterName  # initialize an attribute name
        self.size = monsterSize  # initialize an attribute size
        self.location = monsterLocation  # initialize an attribute location

    # Overriding method __str__ to print "monsterName" attacks "location"
    def __str__(self):
        return self.name + " attacks " + str(self.location)

    # Overriding method __eq__ to compare monster name and size
    def __eq__(self, other):
        return self.name == other.name and self.size == other.size

    # Overriding method __add__ to concatenate two monster names
    def __add__(self, other):
        return self.name + str(other.name)


# Creating instances Dougie and Pluto
LochNess = Monster("LochNess", 4, "London")
BigFoot = Monster("BigFoot", 5, "Tokyo")

# Testing the print method
print()
print(LochNess)
print(BigFoot)
print(LochNess == BigFoot)
print(LochNess + BigFoot)
```

```
LochNess attacks London
BigFoot attacks Tokyo
False
LochNessBigFoot
```

```
import sys


def main():
    # Gather input from the user
    data = input('Enter a string: ')

    # variables to keep track of string length using count
    # and a new str variable to hold the reverse string
    count = 0
    count1 = len(data)
    newdata = ""

    # call the strlenrecursion function to print the string
    # and length each time the first character of the
    # string is removed until the string is empty
    # using a recursive method
    strlenrecursion(data, count)
    print()

    # call the revstrrecursion to print the string in
    # reverse order by using a recursive method
    print("Reverse the String")
    revstrrecursion(data, newdata, count1)

    # first remove punctuations from the string and
    # then pass the updated string to the ispalindrome
    # function to check to see if the string is a
    # palindrome
    data = (removePunctuation(data))
    print()
    print("Is the string", data, "a palindrome?")
    print(ispalindrome(data))


# this function takes a string and int count as a parameter
# and is recursively called to remove a letter and keep
# track of how many letters were removed
# the recursive call ends when it reaches the base case
# when the string length equals 0
def strlenrecursion(data, count):
    if(len(data)) == 0:
        print("letters removed: ", count)
        return
    else:
        print(data, " letters removed: ", count)
        count += 1
        strlenrecursion(data[1:], count)
```

```
# This function takes two string and an int count parameter
# and is recursively called to obtain the reverse of a string.
# The new string takes in the last character of the original
# string and then the original string has its last character
# removed. The count starts with the length of the string
# and decrements until the original string is empty.
# When the count reaches 0, it has entered the base case and
# it returns the new reversed string
def revstrrecursion(data, newdata, count1):
    if count1 == 0:
        print(newdata)
        return
    else:
        count1 -= 1
        newdata += data[-1]
        data = data[:-1]
        revstrrecursion(data, newdata, count1)


# This function removes any punctuation that a string may have
def removePunctuation(data):
    data = ''.join(c if c.isalpha() else '' for c in data.lower())
    return data


# This function recursively calls itself to check if the first
# and last index matches. Each time it enters into the recursive
# call, the starting and ending index of the original string
# decreases.
def ispalindrome(data):
    if len(data) < 2:
        return True
    if data[0] != data[-1]:
        return False
    return ispalindrome(data[1:-1])


if __name__ == "__main__":
    sys.exit(main())
```

```c
#include <stdio.h>

int x = 1; // global variable

// This function accepts a pointer variable to the global
// variable as a parameter and returns the pointer variable
int MyFunction(int *x)
{
    *x += 1; // Increments the value of the pointer variable by 1
    return *x;
}

int main(void)
{
    // Assign a pointer variable to the global variable
    int *ptrGlobal = &x;

    // Output statements to reflect that the global variable
    // has been updated when passed by reference to MyFunction
    printf("Observing no referential transparency \n");
    printf("where the global int variable is equal to 1\n");
    printf("\nFirst call to MyFunction\n");
    printf("%d\n", MyFunction(&x));
    printf("Second call to MyFunction\n");
    printf("%d", MyFunction(&x));

}
```