Solutions

1. Write a C program that receives 10 float numbers from the console and sort them in nonascending order, and prints the result

#include <stdio.h>

int main() //Program Start
{

    float numbers[10],result; //definition of float variables and numbers[10] is a array
    int i,p,k; //definition of integer variables

    printf("The program take 10 float numbers and organize in decreasing order.\n\n");

    //print message to enter 10 numbers
    printf("enter 10 numbers:(Note:enter one number and press enter)\n\n");

    for(i=0;i<10;i++) //loop for input
        scanf("%f",&numbers[i]); //the input numbers are save in numbers[i]

    //print message of the numbers the user enter
    printf("Numbers that you enter are:\n\n%f\n%f\n%f\n%f\n%f\n%f\n%f\n%f\n%f\n%f\n",numbers[0],numbers[1],numbers[2],numbers[3],numbers[4],numbers[5],numbers[6],numbers[7],
        numbers[8],numbers[9]);

    //print a message of the numbers in decreasing order
    printf("decreasing order:\n\n");

    for(i=10;i>=0;i--) //loop for compare the numbers enter
    {
        for(k=10;k>=0;k--) //loop for compare the numbers with the array numbers[k]
        {
            if(numbers[i]<numbers[k]) //compare
                
                

result = numbers[i]; // save number

numbers[i] = numbers[k]; // save the greater number one after another
numbers[k] = result; }

// print the numbers in decreasing number
printf(" %f
n %f
n %f
n %f
n %f
n %f
n %f
n %f
n %f
n %f
n %fn", numbers[0], numbers[1], numbers[2], numbers[3], numbers[4], numbers[5], numbers[6], numbers[7], numbers[8], numbers[9]);

// Message to exit the program
printf("Press any button and enter\n");
scanf("%x", &p);

} // end of the program

---

2. Write a C program that receives an integer number and determines if it is a prime number or not.

/* Question 2: Write a C program that receives an integer number and determines if it is a prime number or not*/

#include <stdio.h>
#include <stdlib.h>
#define FALSE 0
#define TRUE 1

int main(void)
{
    int num1; // initialize variables

    printf("Please enter an integer number that is desired to be prime-tested:\n"); // User Input
scanf("%d", &num1); // Stores Inputted value in num1 location

printf("The number you just entered was %d\n\n", num1); // Reminds the user the number they just entered

printf("1 determines that the inputted number is Prime.\n");

printf("0 determines that the inputted number is NOT Prime.\n\n %d", prime_test(num1)); // Outputs the result of the called function

getch(); // Same as system Pause

}

int prime_test(int n) // Starts next function
{
    long i; // Initializes i

    if (n < 2) return FALSE; // Prime numbers are being assumed greater than 2 (3, 7, 11, etc. 1 and zero are excluded.)

    for (i = 2; i < n; i++) // Starts loop
    {
        if ((n % i) == 0) return FALSE; // If remainder of division is zero, then it is not a prime number
    }

    return TRUE; // If conditions are not met, then the number has to be prime, // as its only divisible by 1 and by itself.

}

3. Write a C program that receives 11 float numbers of three significant figures from the console. Sort it in non-ascending order and print the midpoint of the list.
```c
#include <stdio.h>
#include <stdlib.h>

main (void)
{
    float Number [11],z1,z2; // Variables
    int index,x;
    printf("Write 11 float numbers:
"); // Asking for a float number
    for (index=0; index<11 ; index++)
    {
        printf("Write a number \
",index);
        scanf("%f",&Number [index]);
    }

    for (x=0; x<11; x++) //This part of the code is to sort the 11 numbers in a non ascending form
    {
        for (index=0; index<10-x; index++)
        {
            if (Number [index] >Number[index+1])
            {
                z1=Number [index];
                z2=Number [index+1];
            }
            Number [index]=z1;
            Number [index+1]=z2;
        }
    }
```
printf("Middle Point is %f\n", Number[5]); // In this instruction it show you the middle point
system("PAUSE");
return 0;

4. Write a C program that takes as input two memory addresses between 0 and \((2^8) - 1\) and prints the displacement address (e.g. the distance between two points).

/* This code can be executed using a Dev-C++ compiler.

Question 4: Write a C program that takes as input two memory addresses between 0 and \((2^8) - 1\) and prints the displacement address (e.g. the distance between two points).

*/
#include <stdio.h>
#include <stdlib.h>

int main(int argc, char *argv[])
{
    int address1 = -1; // It is initialized to -1 so we can use it as a condition for the while loop.
    int address2 = -1;
    int displacement; // This will hold the distance between the addresses.

    // This while loop will assure the program to get the correct input we desire for the first address.
    while(address1 < 0 || address1 > 255) // The input must be bigger than 0 and less than 256 so that we can get out the while loop.
    {
    }
printf("Enter first address: ");
scanf("%d", &address1);
}

// This while loop will assure the program to get the correct input we desire for the second address.
while(address2 < 0 || address2 > 255)// The input must be bigger than 0 and less than 256 so that we can get out the while loop.
{
    printf("Enter second address: ");
    scanf("%d", &address2);
}

printf("Address1: %d \n", address1);
printf("Address2: %d \n", address2);

displacement = fabs(address1 - address2); // This computes the distance of the addresses by subtracting them and using absolute value.
printf("Displacement: %d \n", displacement);
system("PAUSE");
return 0;

5. Write a C program that receives as input 4 parameters named x1 ,x2, y1 ,y2 representing continuous points of a linear equation(f(x)=x*m+b). Calculate the rate of change and also indicate if it is increasing or decreasing.

#include<stdio.h>
#include<conio.h>

void main()
{

int x1,x2,y1,y2;
float m; //Initialize Parameters

printf("Please enter first X parameter.\n"); //Ask user for parameters
scanf("%d",&x1);

printf("Please enter corresponding Y parameter.\n");
scanf("%d",&y1);

printf("Please enter second X parameter.\n");
scanf("%d",&x2);

printf("Please enter corresponding Y parameter.\n");
scanf("%d",&y2);

m = (y2-y1)/(x2-x1); //Calculate rate of change

printf("The rate of change is:\n\n",m); //Display rate of change

if(m > 0){ //Display if it is increasing, decreasing or neither
    printf("The equation is increasing.\n");
}
else if(m < 0){
    printf("The equation is decreasing.\n");
}
else if (m == 0) {
printf("The equation is not changing. \n");

}

system("PAUSE");
return 0;
}

6. Write a C program that reads from the console a random number and determine if it is a power of 2.

#include <stdio.h>

int main () {
    int x;
    //This line prompts the user to write a number
    printf("Please enter a number\n");
    scanf("%d", &x);
    //This function checks whether the number is a power
    //of 2 or not
    if((x&(x-1))==0){
        printf("The number you selected is a power of 2\n");
    }
    else {
        printf("The number you selected is not a power of 2\n");
    }
    system("PAUSE");
    return 0;
}
7. Write a C program that receives a positive float number and divides it by two until the result is less than or equal to zero. Print the number of iterations required to meet the specification. Explain in the comments the behavior of your code.

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

int main(int argc, char *argv[])
{
    float x;
    printf("Enter a floating point number: ");
    scanf("%f", &x); //Asks the user for a positive floating point number.
    if(x <= 0) {
        //If the input is negative, throws an exception.
        printf("Invalid response!\n");
        system("PAUSE");
        return 0;
    }
    else {
        int count = 0; //To count the number of iterations.
        float t;

        while(!(x<=0)){ //While the answer is not equal or less than zero.
            t = x/2;
            x = t;
            count++;
        }
        printf("Number of iterations: %d\n", count);
    }
}
```
x=t;     //Updates by the value of the division by 2.

count++; //Counts after each division iteration.
}

printf("Number of iterations it took:%d\n ", count); //Number of iterations.

system("PAUSE");

return 0;}
}

8. Write a C program that takes as input an 8-bit number (e.g. 11011011). Perform and print the logical NOT of the number. (Use the bitwise operations).

/*Question 8: Write a C program that takes as input an 8-bit number (e.g. 11011011). Perform and print the logical NOT of the number. (Use the bitwise operations.*)*/

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

int main()
{
    //Initializing Variables:
    int binary_array[8];
    int notbinaryarray[8];
    int bincountr = 0;
    int bincountr2 = 0;
    int r,i;
printf("Enter desired binary number (from Most Significant Bit\nto Least Significant Bit):\n");
for ( r = 0; bincountr < 8; r++)  //Binary number loop
{
    scanf("%d",&binary_array[r]);
    bincountr++;
}

printf("The inputted binary number was: \n");
printf("%d%d%d%d%d%d%d%d\n", binary_array[0],binary_array[1],binary_array[2],
//Displays inputted binary value
binary_array[3],binary_array[4],binary_array[5],binary_array[6],binary_array[7]);
printf("The logical NOT operation of the inputted number is:\n");

for ( i = 0; bincountr2 < 8; i++)  //Calculates logical NOT bit by bit using bitwise operation XOR
{
    notbinaryarray[i]=(binary_array[i] ^ 1 ) ;
    bincountr2++;
}

printf("%d%d%d%d%d%d%d%d\n", notbinaryarray[0],notbinaryarray[1],notbinaryarray[2],
//Displays logical NOT of inputted value.
notbinaryarray[3],notbinaryarray[4],notbinaryarray[5],notbinaryarray[6],notbinaryarray[7]);
printf("\n");
getch();  //Another method of writing System Pause in dev c++
return 0;

9. Write a C program that takes as input two 8-bit numbers and calculate the AND, OR, and
XOR logical representations. (Use the bitwise operations).

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

//Question 9: Write a C program that takes as input two 8-bit numbers and calculate the AND, OR and XOR logical representations. (Use the bitwise operations)

main (void) //Program Start
{
    int number1 [8]; //First binary number of 8-bits
    int number2 [8]; //Second binary number of 8-bits

    int i; //Variable Initialization

    printf("Enter the first binary number of 8-bits:\n");

    for (i=0; i<8; i++) //Asking for the first binary number
    {
        scanf("%d", &number1 [i]);
    }

    printf("Enter the second binary number of 8-bits:\n");
for (i=0; i<8; i++)                  //Asking for the second binary number
{
    scanf ("%d", &number2 [i]);
}

printf ("The AND logical representation of the two entered binary numbers is:\n");

for (i=0; i<8; i++)                  //AND logical representation using "&" like AND operator
{
    printf ("%d\n", number1 [i] & number2 [i]);
}

printf ("The OR logical representation of the two entered binary numbers is:\n");

for (i=0; i<8; i++)                 //OR logical representation using "|" like OR operator
{
    printf ("%d\n", number1 [i] | number2 [i]);
}

printf ("The XOR logical representation of the two entered binary numbers is:\n");

for (i=0; i<8; i++)                //XOR logical representation using "^" like XOR operator
{
    printf ("%d\n", number1 [i] ^ number2 [i]);
}
10. Write a C program that takes as input an 8-bit number. Perform and print the logical NEG of this number. (Use the bitwise operations).

/*
This code can be executed with a Dev-C++ compiler.

Question 10: Write a C program that takes as input an 8-bit number.
Perform and print the logical NEG of this number. (Use the bitwise operations.)
*/

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

int main(int argc, char *argv[])
{
    int binary[8]; // Holds the 8-bit binary number
    int i; // Counter for the for loops

    printf("Enter an 8-bit (Only 0 and 1 are allowed as input)\n");

    // This for loop will run 8 times to get as input each bit
    for(i=0; i<8; i++)
    {
        
        system("PAUSE"); //Screen pause after the program runs

    } //Program End

} //Program End
int bit = -1; // Holds the input from the user. I initialized to -1 to use it also as a flag to check that the input is only 0 or 1.

while(bit == -1)// This while loop only permits 0 and 1 as inputs for the array.
{
    scanf("%d", &bit);
    if(bit != 0 && bit != 1)
        bit = -1;
}

binary[i] = bit;

printf("Your binary number is: ");

// This for loop prints the 8-bit binary number received.
for(i = 0; i<8; i++)
{
    printf("%d", binary[i]);
}

/*
   This for loop actually does the real job of negating the binary number.
   Each iteration will check what is the bit in the specified index (i), if
   the bit is 0 then it will be changed to 1. If we have a 1 then it will be changed to 0.
*/
for(i=0; i<8; i++)
{


11. Write a C program that takes as input two 8-bit binary numbers and perform Binary ADD and print the result. (Use the bitwise operations).

```c
#include <stdio.h>
#include <stdlib.h>
#define SIZE 8

if(binary[i]==0)
    binary[i] = 1;
else
    binary[i]=0;

printf("\nApplying the NEG operator to the binary number results: ");

// This for loop prints the negated 8-bit binary number received.
for(i = 0; i<8; i++)
{
    printf("%d", binary[i]);
}

printf("\n");

system("PAUSE");
return 0;
}
```
int main(int argc, char *argv[]) {
    int i, key; // defining all variables and two arrays
    int f[SIZE];
    int s[SIZE];

    printf("Enter the first 8-digit binary number\n\n");

    for(i=SIZE-1; i>=0; i--) // this loop assigns the input to a location in the array f
    {
        scanf("%d", &key);
        f[i] = key;
    }

    printf("\n\nEnter the second 8-digit binary number\n");

    for (i=SIZE-1; i>=0; i--) // this loop assigns the input to a location in the array s
    {
        printf("%d", f[i]);
    }

    printf("\nEnter the second 8-digit binary number\n");

    for (i=SIZE-1; i>=0; i--) // this loop assigns the input to a location in the array s
    {
        scanf("%d", &key);
        s[i] = key;
    }

    for(i=SIZE-1; i>=0; i--)
    {
        printf("%d", s[i]);
    }
}

int main(int argc, char *argv[])
scanf("%d",&key);
s[i] = key;
}
for(i=SIZE-1; i>=0; i--)
{
    printf("%d", s[i]);
}

int sum[SIZE+1];
int a=0;
int j, sum1;

for(j=0; j<SIZE; j++)//This loop executes the sum of the binary numbers and assigns the carry to a variable a
{
    sum1 = f[j]+s[j] + a; //when the som is 2 the carry a is equal to 1 and this location im the array is 0
    if(sum1 == 2)
    {
        a = 1;
        sum[j] = 0;
    }
    else if(sum1 == 3)
    {
        sum[j] = 1;
        a = 1;
else if(sum1 < 2)
{
    a = 0;
    sum[j] = sum1;
}

if(j==SIZE-1)
{
    sum[SIZE]= a;
}

printf("\nThe ADD: ");
for(i=SIZE; i>=0; i--)
{
    printf("%d", sum[i]);
}

system("PAUSE");
return 0;

12. Write a C program that takes as input an 8-bit number and calculates the 2’s complement logical representation. (Use the bitwise operations).

#include <stdio.h>
int main (int argc, char * const argv[]) {

    //This is an array of 8 integers called digits
    int digits[8];
    int i;
    int n;

    printf("Please enter an 8 bit number, the first number being 2^0 and so on until 2^7:\n");

    //In this part the scanned numbers received by the user are
    //arranged into each space in the array from 2^0 to 2^7
    scanf("%d\n", &digits[0]);
    scanf("%d\n", &digits[1]);
    scanf("%d\n", &digits[2]);
    scanf("%d\n", &digits[3]);
    scanf("%d\n", &digits[4]);
    scanf("%d\n", &digits[5]);
    scanf("%d\n", &digits[6]);
    scanf("%d", &digits[7]);

    //Here it starts to create a loop searching
    //from the first number to the last
    //checking whether it has to be a 1 or 0 or stay the same
    for(i=1; i<8; i++)
    {
        if(digits[0]==1 && digits[i]==0)
        {
            digits[i]=1;
        }
    }
else if(digits[0]==1 && digits[i]==1){

digits[i]=0;
}

else if(digits[0]==0 && digits[i]==1){

for(n=1; n<8; n++){

if(digits[i+n]==1){

    digits[i+n]=0;
}

else if(digits[i+n]==0){

    digits[i+n]=1;
}
}
}

//The numbers are printed in their 2's compliment format

printf("The numbers 2's compliment is:\n");

printf("%d\n", digits[0]);

printf("%d\n", digits[1]);

printf("%d\n", digits[2]);

printf("%d\n", digits[3]);

printf("%d\n", digits[4]);

printf("%d\n", digits[5]);

printf("%d\n", digits[6]);

printf("%d\n", digits[7]);
system("PAUSE");

return 0;
}

13. Write a C program that generates a random positive float number in a fixed interval with a
maximum distance between endpoints of 100 (e.g. [0,100] or [101,201]). Both interval points are
given
by the user through console. The generated random number inside the given interval must be less
than
or equal to 10975. Calculate the factorial (n!) of the resulting random number.

/*
This code can be executed with a Dev-C++ compiler.

Question 13: Write a C program that generates a random positive float number
in a fixed interval with a maximum distance between endpoints of
100 (e.g. [0,100] or [101,201]). Both interval points are given by
the user through console. The generated random number inside the given
interval must be less than or equal to 10975. Calculate the factorial (n!)
of the resulting random number.

*/

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

int main(int argc, char *argv[])
{ 
    int flag = 0; // This flag is used to control the while loops to get the correct input from the user.
    int i; // This is the counter for the for loops.
    double factorial = 1; // This will accumulate the factorial for our random number.
    int endpoint1, endpoint2; // This will be the endpoints for the interval that our random number will be inside of.
    int distance; // This will be the distance between the two endpoints
    srand(time(NULL)); // This will help generate better random numbers by using the current time as the seed.

    // This while loop will assure the program that the user only inputs a number less than 10875.
    while(flag == 0)
    {
        printf("Enter first endpoint of interval: ");
        scanf("%d", &endpoint1);

        if(endpoint1 <= 10875 && endpoint1 >= 1) // It makes sure that the number is smaller than 10876 and bigger than 0.
            flag = 1; // When this flag changes to 1, the user's input was correct.
    }

    flag = 0; // Since we will used it again for our second input

    // This while loop will assure the program that the user only inputs a number that will have a maximum distance of 100 from the first endpoint.
    while(flag == 0)
printf("Enter second endpoint of interval: ");
scanf("%d", &endpoint2);

distance = endpoint2 - endpoint1;

if(distance <= 100 && distance >= 1) // It makes sure that the distance between endpoints is smaller
    // than 101 and bigger than 0.
    flag = 1; // When this flag changes to 1, the user's input was correct.
}

printf("The inputs for the interval are [ %d , %d ]\n", endpoint1, endpoint2);

int random = endpoint1 + (rand()%distance)+1; // This generates our random number between our two endpoints.

printf("Random number: %d\n", random);

/*
This for loop accumulates the factorial of each number until we reach our random number.
Unfortunately I couldn't find a data type that could hold up such big numbers when it passes over 170.
The unsigned long int can only hold up to 4294967295. The one I used was double which holds up to
1.8x10^308.

The more appropriate datatype would be long double which can hold up to 1.2x10^4932.
*/
for (i = 1; i <= random; i++)
{
    factorial *= i;
}

printf("The factorial for the random number %d is: %f
", random, factorial);

system("PAUSE");
return 0;

14. Assume an ideal voltage divider circuit. Write a C program that reads from the console a
parameter float (Vin) and a second parameter float (Vout) and print the load resistor needed for this
design with its unit. Fix the other resistor to 1kOhm.

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

//Question 14: Assume an ideal voltage divider circuit.
//Write a C program that reads from the console a parameter float (Vin)
//and a second parameter float (Vout) and print the load resistor needed
//for this design with its unit. Fix the other resistor to 1kOhm.

int main(int argc, char *argv[])
{
    float Vout; //Declaring float variables output and input voltages
    float Vin;
    float R1 = 1000; //asigning the fixed resistor to 1 Kohm

printf("Please enter the input voltage\n");
scanf("%f",&Vin);

printf("Enter the output voltage\n");
scanf("%f",&Vout);

float Rload;//declaring variable that is being calculated

Rload = (-R1*Vout)/(Vout-Vin);

printf("The Load resistor needed for this design is %f\n",Rload);

system("PAUSE");
return 0;

15. Write a C program that takes as input a positive fractional number and converts it to binary.

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

main (void)             //Program Start
{
    float Num;             //Variables initialization. In this code "float" is used for the fraction division and "int" is used for the other variables and parameters.
    float Denom;
    int i;
    int Number1;

    //Program code goes here.
}
```c
int n=0;
int Numb[n];
int j;
int x=0;

printf("Please enter the positive fraction numerator:\n");
scanf("%f", &Num); //Asking for the numerator for the fraction division

printf("Please enter the positive fraction denominator:\n");
scanf("%f", &Denom); //Asking for the denominator for the fraction division

Num = Num/Denom; //Numerator and denominator division
i = Num; //The division result is assigned to the "i" variable
Number1 = i; //The value of "i" is assigned to the variable "Number1"
Denom = Num-i; //The decimal side of the number is obtained

do
{
    Numb[n]= Number1%2; //Division remainder (residue)
    Number1 = Number1/2; //New value divided by 2
    n++; //The value of "n" is increasing during the "do-while" is running
}

while(Number1 != 0 ); //Comparing the number with "0"
```
printf("\n");

for(i=n-1; i>=0; i--)         //Reversing the numbers order
{
    printf("%d", Numb[i]);
}

printf(".");                  //Point between the integer and decimal part of the number

Num = Denom;                  //Decimal value is assigned to the "Demon" variable

do
{
    Num = Num*2;           //Multiply by 2 for reached the 15 values
    j=Num;               //Comparation

    if(j>0)               //If the "j" value is greater a "1" is printed because a
    {
        //Multiplication by 2 give a result with a integer part and
        printf("1");    //With a decimal part.
        Num = Num-j;
    }

    else                  //If the result of the multiplication dont give a integer
    printf("0");        //Part or a decimal part a "0" is printed.
    x++;                //Increment x
while(x!=15);                      //The maximum quantity of values for a binary number is 15

printf("\n");

system ("PAUSE");                //Pausing the result screen after the program run

}                                 //Program End

16. Write a C program that takes a non-negative integer number as input and prints its binary representation.

/*Assume the output number is an 8 bit binary number.*/

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

int main()
{
    int nose=1;                     //initializing variables and matrix
    int inputnumber;
    int binaryoutput[8];           //Range of unsigned values: 0 - 255
    int i;
    int n;

    printf("Enter the desired integer to be converted to binary (enter -1 to finish)\n");
scanf("%d", &inputnumber);         //User input

while( inputnumber != -1 )        //While loop to input different numbers at one's own convenience
{
    if(inputnumber < 256 && inputnumber >= 0)  //restrictions taken from the Range of values possible
    {
        for( i = 0; i < 8; i++ )
        {
            n = inputnumber % 2;
            if( inputnumber == 0 )
            {
                binaryoutput[i]=0;
            }
            else
            {
                binaryoutput[i]=n;
            }
        }
        inputnumber = inputnumber / 2;
    }
    else
    {
    }
}

else
{
}
printf("\nNumber out of boundary please re-enter to end -1\n"); //Wrong input gives a chance to repeat.

}  
printf("\nThe binary representation is:\n"%d%d%d%d%d%d%d \n",binaryoutput[7],binaryoutput[6],binaryoutput[5],
      binaryoutput[4],binaryoutput[3],binaryoutput[2],binaryoutput[1],binaryoutput[0] );

printf("\nEnter the desired integer to be converted to binary\n");
scanf("%d", &inputnumber);
}
system("PAUSE");
return 0;
}

17. Write a C program that takes a 2 digit hexadecimal number from the user through console and display it in binary format.

#include <stdio.h>

int main (int argc, char * const argv[]) {
    // In this part we create an array of 2 characters
    char number[2];
    int i;
    printf("Please enter a 2 digit hexadecimal number\n");
    // Here it scans the number given and stores it in the array
    scanf("%s", number);
    // It starts to create a loop from 0 to 2 checking each number
    for(i=0; i<2; i++){
// If the number is equal to one number in hexadecimal it print in
// the binary format
    if(number[i] == '0'){
        printf("0000");
    }
    else if(number[i] == '1'){
        printf("0001");
    }
    else if(number[i] == '2'){
        printf("0010");
    }
    else if(number[i] == '3'){
        printf("0011");
    }
    else if(number[i] == '4'){
        printf("0100");
    }
    else if(number[i] == '5'){
        printf("0101");
    }
    else if(number[i] == '6'){
        printf("0110");
    }
    else if(number[i] == '7'){
        printf("0111");
    }
} else if(number[i] == '8'){
    printf("1000");
}
else if(number[i] == '9'){
    printf("1001");
}
else if(number[i] == 'A' || number[i] == 'a'){
    printf("1010");
}
else if(number[i] == 'B' || number[i] == 'b'){
    printf("1011");
}
else if(number[i] == 'C' || number[i] == 'c'){
    printf("1100");
}
else if(number[i] == 'D' || number[i] == 'd'){
    printf("1101");
}
else if(number[i] == 'E' || number[i] == 'e'){
    printf("1110");
}
else if(number[i] == 'F' || number[i] == 'f'){
    printf("1111");
}
else {
    printf( "You did not follow the rules\n" );
}

printf( "\n" );

system("PAUSE");
return 0;
}

18. Write a C program that takes an 8-bit number through console as input. Convert it to hexadecimal format and display it as output.

#include <stdio.h>
#include <string.h>

int bin2dec(char *bin); // Declares Function With Characters
int main() // Main Program Starts
{
    char bin[80] = ""; // Declares Character Array

    char *p; // Declares Character Pointer

    int dec; // Saves Variable Name Into Memory

    while(strcmp(bin,"0")) // Start While Loop With String Compare Condition

{ 

    printf("\nPlease Enter an 8-Bit Binary Number (Press 0 To EXIT): ");      // Prompts For User Input

    fgets(bin, sizeof(bin), stdin);                                        // Reads Characters From Stream And Stores Them
As A C String

    // Until (num-1) Characters Have Been Read Or Either a A
    // Newline Or A The End-of-File Is Reached, Whichever

    if ((p = strchr(bin,\n')) != NULL)                                       // If Variable P Equals A String Character
Condition...

    {
        *p = \0';                                                               // If False Condition Declare Null
    }

    dec = bin2dec(bin);

    if (dec) printf("\n\nDecimal = %d
Hexadecimal = 0x%04X\n",dec,dec);      // Displays Decimal And
Hexadecimal Values

    } 

    getchar();                                                               // Pauses Program

    return 0;                                                                 // Return To Main
int bin2dec(char *bin)
{
    int b, k, n;                                        // States Multiple Variables
    int len, sum = 0;                                    // States Variable Into Memory

    len = strlen(bin) - 1;

    for(k = 0; k <= len; k++)                           // Initialize For Loop With Counter
    {
        b = 1;
        n = (bin[k] - '0');                              // Char To Numeric Value
        if ((n > 1) || (n < 0))                           // Makes Sure It Is A Binary Input
        {
            puts("\n\nERROR! BINARY has only 1 and 0!\n\n"); // Displays Error Message If Condition Is Not Met
            return (0);
        }
        b = b << (len-k);
        sum = sum + n * b;                                // Sums All The Binary Positions
        printf("\n+n%d*%d ",n,b);                           // Displays Process
    }
    return(sum);                                        // Returns To Sum Loop
}