C Coding Practice Question

Q.1. A Company wants to organize the data of their employees. Help them with a C program to accept the details of employee and display them using structure. Details consist of Employee ID, Name, Designation, Department, Salary.

#Solution

```
#include<stdio.h>
struct employee
{
    int id;
    char name[20];
    char designation[20];
    char dept[20];
    int sal;
};
void main()
{
    struct employee emp;

    scanf("%d",&emp.id);
    scanf("%s",emp.name);
    scanf("%s",emp.designation);
    scanf("%s",emp.dept);
    scanf("%d",&emp.sal);

    printf("Employee-Id: %d\n",emp.id);
    printf("Name: %s\n",emp.name);
    printf("Designation: %s\n",emp.designation);
    printf("Department: %s\n",emp.dept);
    printf("Salary: %d",emp.sal);
}
```

Q.2. Write a program in C to print first n natural numbers using recursion.

#Solution

```
#include<stdio.h>
int  numPrint(int, int);
int main()
{
    int n = 1;
    int n1;
    scanf("%d",&n1);
    numPrint(n,n1);
    return 0;
}
int numPrint(int n,int n1)
{
    if(n<=n1)
    {
         printf("%d ",n);
         numPrint(n+1,n1);
    }
}
```

Q.3. Write a 'C' program to accept customer details such as: Account_no, Name, Balance using structure. Assume 3 customers in the bank. Write a function to print the account no. and name of each customer whose balance < 1000 Rs, if no customer found whose balance is less than 1000, print None.

#Solution

```
#include<stdio.h>

struct bank

{

    int acc_no;

    char name[20];

    int bal;

}b[3];


void check(struct bank b[],int n)

{

    int i,c=0;

    for(i=0;i<n;i++)

    {

        if(b[i].bal<1000)

        {

            printf("%d\n",b[i].acc_no);

            printf("%s\n",b[i].name);

            printf("%d\n",b[i].bal);

            c++;

        }

    }

   if (c==0)

   {
        printf("None");
    }
}
int main()
{
    int i;
    for(i=0;i<3;i++)
    {
        scanf("%d",&b[i].acc_no);
        scanf("%s",b[i].name);
        scanf("%d",&b[i].bal);
    }
    check(b,3);
    return 0;
}
```

Q4. You are transporting n numbers boxes through a tunnel, where each box is parallelepiped, and is characterized by its length, width and height.

The height of the tunnel feet and the width can be assumed to be infinite. A box can be carried through the tunnel only if its height is strictly less than the tunnel's height h. Find the volume of each box that can be successfully transported to the other end of the tunnel. Note: Boxes cannot be rotated.

Input: take two integer n and h.

n line have length, width and height of nth -1 box.

constraints:

1 <=n,h,length,width and height,height <= 100

Solution :

```
#include <stdio.h>
#include <stdlib.h>
#define MAX_HEIGHT 41

struct box
```

```
{
int length;
    int width;
    int height;
};

typedef struct box box;

int get_volume(box b) {
    int volume;
    volume=(b.length)*(b.width)*(b.height);
    return volume;
}

int is_lower_than_max_height(box b) {
    int result = b.height < MAX_HEIGHT ? 1:0;
    return result;
}

int main()
{
int n,ma;
scanf("%d%d", &n,&MAX_HEIGHT);
box *boxes = malloc(n * sizeof(box));
for (int i = 0; i < n; i++) {
		scanf("%d%d%d", &boxes[i].length, &boxes[i].width, &boxes[i].height);
}
for (int i = 0; i < n; i++) {
		if (is_lower_than_max_height(boxes[i])) {
				printf("%d\n", get_volume(boxes[i]));
		}
}
return 0;
}
```

Q4. Imagine you're a mathematician studying the properties of numbers. You come across a unique number n and its reverse r, such that when the number is raised to the power of its reverse, a fascinating result occurs. You have given a number and its reverse. You have to determine what the result of this calculation would be?

As answers can be very large, print the result modulo 10^9 + 7.

Solution :

```
#include <stdio.h>
# define mod 1000000007
long long power(int N,int R)
{
    if(R==0) return 1;
    long long temp = power(N,R/2);
    temp=(temp*temp)%mod;
    if(R%2==1)
    {
        return (temp*N)%mod;
    }
    return temp;
}
int main()
{
    long long N,R;
    scanf("%ld%ld",&N,&R);
    long long ans =power(N,R);
    printf("%ld",ans);
}
```

Q6 Imagine you are a librarian tasked with sorting a large collection of books in ascending order by book number.To accomplish this, you decide to use a selection sort algorithm implemented through recursion. Starting with the first book, you compare it to each subsequent book and swap their positions if necessary, continuing this process until the entire collection is sorted. By using recursion, you are able to break down the sorting process into smaller, more manageable steps, which helps to simplify the overall task of organizing the library's book collection.

Solution:

```
#include <stdio.h>


void selection(int [], int, int, int, int);
```

```
int main()
{
        int list[30], size, temp, i, j;

        printf("Enter the size of the list: ");
        scanf("%d", &size);
        printf("Enter the elements in list:\n");
        for (i = 0; i < size; i++)
        {
        scanf("%d", &list[i]);
        }
        selection(list, 0, 0, size, 1);
        printf("The sorted list in ascending order is\n");
        for (i = 0; i < size; i++)
        {
        printf("%d  ", list[i]);
        }

        return 0;
}

void selection(int list[], int i, int j, int size, int flag)
{
        int temp;

        if (i < size - 1)
        {
        if (flag)
        {
        j = i + 1;
        }
```

```
            if (j < size)
            {
            if (list[i] > list[j])
            {
            temp = list[i];
            list[i] = list[j];
            list[j] = temp;
            }
            selection(list, i, j + 1, size, 0);
            }
            selection(list, i + 1, 0, size, 1);
            }
}
```

Q7.Given two integers n and r, find nCr. Since the answer may be very large, calculate the answer modulo 109+7.

Constraints:

1 ≤ n ≤ 30

1 ≤ r ≤ 30

char will contain upper and lower case alphabets.

Solution :

```
#include <stdio.h>
#define MOD 1000000007
int nCr(int n, int r) {
    if (r == 0 || r == n)
        return 1;

    int ans = (nCr(n-1, r-1) % MOD + nCr(n-1, r) % MOD) % MOD;
    return ans;
}
int main() {
    int n,r;
    scanf("%d%d",&n,&r);
    int ans = nCr(n, r);
    printf("%d",ans);
    return 0;
}
```

Q8. You are a programmer who has been tasked with writing a C program to find the nth Fibonacci number using recursion. Your boss wants the program to be able to handle large values of n without crashing or slowing down too much. How would you approach this task?

Constraints:

0<= n <= 30

Solution :

```
#include <stdio.h>
int fibonacci(int num) {
    if (num == 0 || num == 1) {
        return num;
    } else {
        return fibonacci(num - 1) + fibonacci(num - 2);
    }
}
int main() {
    int num;
    scanf("%d", &num);
    printf("%d\n",fibonacci(num));
    return 0;
}
```

Q9 **I work as a network administrator at a large corporation, and one of my responsibilities is to monitor the network traffic and identify any potential security threats. To do this, I need to know the class of each IP address that is connected to the network. Using a program to find the class of an IP address, I am able to quickly determine whether a particular address is a part of a private or public network, which helps me to take appropriate action in case of any security breach**

Solution:

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


void extractIpAddress(unsigned char *sourceString,short *ipAddress)
{
    unsigned short len=0;
    unsigned char  oct[4]={0},cnt=0,cnt1=0,i,buf[5];

    len=strlen(sourceString);
    for(i=0;i<len;i++)
    {
        if(sourceString[i]!='.'){
            buf[cnt++] =sourceString[i];
        }
        if(sourceString[i]=='.' || i==len-1){
            buf[cnt]='\0';
            cnt=0;
            oct[cnt1++]=atoi(buf);
        }
    }
    ipAddress[0]=oct[0];
    ipAddress[1]=oct[1];
    ipAddress[2]=oct[2];
    ipAddress[3]=oct[3];
}

int main()
{
    unsigned char ip[20]={0};
    short ipAddress[4];

    printf("Enter IP Address (xxx.xxx.xxx.xxx format): ");
    scanf("%s",ip);

    extractIpAddress(ip,&ipAddress[0]);

    printf("\nIp
Address: %03d. %03d. %03d. %03d\n",ipAddress[0],ipAddress[1],ipAddress[2],ipAddress[3]);

    if(ipAddress[0]>=0 && ipAddress[0]<=127)
        printf("Class A Ip Address.\n");
    if(ipAddress[0]>127 && ipAddress[0]<191)
        printf("Class B Ip Address.\n");
    if(ipAddress[0]>191 && ipAddress[0]<224)
```

```
        printf("Class C Ip Address.\n");
    if(ipAddress[0]>224 && ipAddress[0]<=239)
        printf("Class D Ip Address.\n");
    if(ipAddress[0]>239)
        printf("Class E Ip Address.\n");

    return 0;
}
```

Q.1. Write a C program to create a student structure having fields student name and student city name. Accept the details of 'n' students, rearrange the data in alphabetical order of student name and display it.

#Solution

```
#include<stdio.h>

#include<string.h>

struct stud

{

    char name[50];

    char city[50];

}s[100];

int main()

{

    struct stud t;

    int i,j,n;

    scanf("%d",&n);

    for(i=0;i<n;i++)

    {

        scanf("%s",s[i].name);

        scanf("%s",s[i].city);
    }

    for(i=0;i<n;i++)
    {
        for(j=i+1;j<n;j++)
        {
            if(strcmp(s[i].name,s[j].name)>0)
            {
                t=s[i];
                s[i]=s[j];
                s[j]=t;
            }
        }
    }

    for(i=0;i<n;i++)
    {
        printf("%s %s\n",s[i].name,s[i].city);
    }
    return 0;
}
```

Q.1. In Fintech Corp. wants to automate their salary calculation, Write a C program to input the details with two employees and calculate the total payment of same employees using structure.

#Solution

```
#include<stdio.h>
struct worker
```

```
{
    char name[20];
    int wage;
    int wdays;
};
void main()
{
    struct worker a,b;

    scanf("%s",a.name);
    scanf("%d",&a.wage);
    scanf("%d",&a.wdays);
    scanf("%s",b.name);
    scanf("%d",&b.wage);
    scanf("%d",&b.wdays);

    int p1=a.wage*a.wdays;
    int p2=b.wage*b.wdays;
    printf("%s %d\n",a.name,p1);
    printf("%s %d\n",b.name,p2);
}
```

Q.2. Write a program in C to calculate the sum of numbers from 1 to n using recursion.

#Solution

```
#include <stdio.h>
int addNumbers(int n);
int main() {
  int num;
  scanf("%d", &num);
  printf("%d", addNumbers(num));
  return 0;
}
int addNumbers(int n) {
  if (n != 0)
    return n + addNumbers(n - 1);
  else
    return n;
}
```

Q.1. Write a program in C to get the largest element of an array using recursion.

#Solution

```
#include<stdio.h>
#define MAX 100
int MaxElem(int []);
int n;

int main()
{
    int arr1[MAX],hstno,i;
    scanf("%d",&n);

    for(i=0;i<n;i++)
        {
                scanf("%d",&arr1[i]);
              }
    hstno=MaxElem(arr1);
    printf("%d",hstno);
    return 0;
}
int MaxElem(int arr1[])
{
    static int i=0,hstno =-9999;
    if(i < n)
    {
         if(hstno<arr1[i])
           hstno=arr1[i];
      i++;
      MaxElem(arr1);
    }
    return hstno;
}
```

Q.1. C program to read information of single student. It contains Name, Roll number, Birthday, admission date. Calculate age of student at the time of admission with using structure.

#Solution

```
#include<stdio.h>
struct student
{
     int roll_num;
     char name[20];
```

```
    struct Date
    {
        int D;
        int M;
        int Y;
    }bd,ad;
};

void main()
{
    int age;
    struct student a;
    scanf("%d",&a.roll_num);
    scanf("%s",a.name);
    scanf("%d-%d-%d",&a.bd.D,&a.bd.M,&a.bd.Y);
    scanf("%d-%d-%d",&a.ad.D,&a.ad.M,&a.ad.Y);
    age=a.ad.Y-a.bd.Y;
    printf("%d Years",age);
}
```

Q.2. Write a program in C to find the sum of digits of a given number using recursion.

#Solution

```
#include <stdio.h>
int DigitSum(int num);
int main()
{
   int n1, sum;
   scanf("%d", &n1);
```

```
    sum = DigitSum(n1);


    printf("%d", sum);


    return 0;
}


int DigitSum(int n1)
{
    if(n1 == 0)
        return 0;


    return ((n1 % 10) + DigitSum(n1 / 10));
}
```

Q.1. Write a program in C to Print Fibonacci Series using recursion, Input number of terms for the Series.

#Solution

```
#include<stdio.h>
int term;
int fibonacci(int prNo, int num);
void main()
{
   static int prNo = 0, num = 1;
   scanf("%d", &term);
   printf("1 ");
```

```
  fibonacci(prNo, num);
}
int fibonacci(int prNo, int num)
{
  static int i = 1;
  int nxtNo;

  if (i == term)
    return (0);
  else
  {
    nxtNo = prNo + num;
    prNo = num;
    num = nxtNo;
    printf("%d ", nxtNo);
    i++;
    fibonacci(prNo, num);
  }
  return (0);
}
```

Q16. Imagine that you are at the base of a staircase that has several steps, numbered from 1 to n. You want to climb to the top of the staircase, but each step has a cost associated with it. You are given an array called "cost" that contains the cost of each step. Specifically, cost[i] represents the cost of climbing the ith step on the staircase.

You have two options for climbing each step. You can either pay the cost of the step and climb one step, or pay the cost of the step and climb two steps. Once you have climbed a step, you cannot go back down.

Your goal is to find the minimum cost required to reach the top of the staircase, which is the nth step
Solution 16:

```
#include <stdio.h>
#include <stdlib.h>
int minCostClimbingStairs(int* cost, int n) {
  int i, oneStep = 0, twoStep = 0;
```

```
    for(i = n-1; i >= 0; i--) {
        int current = cost[i] + ((oneStep < twoStep) ? oneStep : twoStep);
        twoStep = oneStep;
        oneStep = current;
    }
    return (oneStep < twoStep) ? oneStep : twoStep;
}
int main() {
    int n, i;
    scanf("%d", &n);
    int *cost = (int *) malloc(n * sizeof(int));
    for(i = 0; i < n; i++) {
        scanf("%d", &cost[i]);
    }
    int minCost = minCostClimbingStairs(cost, n);
    printf("%d", minCost);
    free(cost);
    return 0;
}
```

Q17. Define a structure named Time to represent the time of day (hour, minute, and second). Write a program to read two times from the user, add them, and display the result on the console.

Solution 17:

```
#include <stdio.h>
struct Time {
    int hour;
    int minute;
    int second;
};
int main() {
    struct Time t1, t2, result;
    scanf("%d%d%d", &t1.hour, &t1.minute, &t1.second);
    scanf("%d%d%d", &t2.hour, &t2.minute, &t2.second);
    result.second = t1.second + t2.second;
    result.minute = t1.minute + t2.minute + result.second/60;
    result.hour = t1.hour + t2.hour + result.minute/60;
    result.second %= 60;
    result.minute %= 60;
    printf("%d:%d:%d\n", result.hour, result.minute, result.second);
    return 0;
}
```

Q18 **Suppose you are working on a project that involves processing large matrices. One of the tasks required is to calculate the sum of non-diagonal elements of a given m x n matrix. Since the size of the matrix can be quite large, you decide to allocate memory dynamically to ensure efficient use of memory. To accomplish this task, you decide to write a 'C' program that takes in the dimensions of the matrix as input and dynamically allocates memory to create the matrix. The program then calculates the sum of all the non-diagonal elements and outputs the result.**

Solution:

```
    #include<stdio.h>
#include<stdlib.h>
int main()
{
    int **a, row,col,i,j,s=0;
    printf("Enter Limit for Rows : ");
    scanf("%d",&row);
    printf("\nEnter Limit for Columns : ");
    scanf("%d",&col);
    a=(int **)malloc(row*sizeof(int*));
    for(i=0;i<row;i++)
    {
        a[i]=(int *)malloc(col*sizeof(int));
    }
    printf("\nEnter Elements for Matrix of Size %d*%d:\n\n",row,col);
    for(i=0;i<row;i++)
    {
        for(j=0;j<col;j++)
        {
            scanf("%d",&a[i][j]);
        }
    }
    printf("\n%d*%d Matrix : \n\n",row,col);
    for(i=0;i<row;i++)
    {
        for(j=0;j<col;j++)
        {
            printf("%3d ",a[i][j]);
        }
        printf("\n");
    }
    printf("\nNon-Diagonal Elements are: ");
    for(i=0;i<row;i++)
    {
        for(j=0;j<col;j++)
        {
            if(i!=j)
```

```
                {
                    printf("%d ",a[i][j]); //Minor diagonal not considered here
                    s=s+a[i][j];
                }
            }
        }
        printf("\n\nSum of Non-Diagonal Elements in Matrix = %d",s);
        return 0;
}
```

Q16. Consider a game where a player can score 2 or 3 or 5 points in a move. Given a total score n, find a number of distinct combinations to reach the given score.

Solution 16:

```
#include <stdio.h>
int countCombinations(int n) {
    if (n == 0) {
        return 1;
    } else if (n < 0) {
        return 0;
    } else {
        return countCombinations(n - 2) + countCombinations(n - 3) + countCombinations(n - 5);
    }
}
int main() {
    int n ;
    scanf("%d",&n);
    int numCombinations = countCombinations(n);
    printf("%d",numCombinations);
    return 0;
}
```

Q17. Create a structure to represent a rectangle with fields for width and height. Write a program that prompts the user to enter the dimensions of two rectangles, calculates the area of each rectangle, and then prints the dimensions and areas of both rectangles.

Solution 16:

```
#include <stdio.h>
struct rectangle {
    int width;
    int height;
};

int main() {
    struct rectangle rect1, rect2;
    int area1, area2;
    scanf("%d%d", &rect1.width, &rect1.height);
    scanf("%d%d", &rect2.width, &rect2.height);
    area1 = rect1.width * rect1.height;
    area2 = rect2.width * rect2.height;

    printf("%d\n",area1);
    printf("%d",area2);

    return 0;
}
```

Q18 **Suppose you are implementing a program to print a stack of books in reverse order, where each book is represented as a node in a linked list. You want to use a stack data structure to keep track of the books as you traverse the linked list. However, the stack is currently in the wrong order, so you need to reverse it before printing. To accomplish this, you decide to write a C program that uses recursion to reverse the stack.**

Solution:

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

struct node
{
    int a;
    struct node *next;
};

void generate(struct node **);
void display(struct node *);
void stack_reverse(struct node **, struct node **);
void delete(struct node **);

int main()
{
    struct node *head = NULL;

    generate(&head);
    printf("\nThe sequence of contents in stack : \n");
    display(head);
    printf("\n\nInversing the contents of the stack\n");
    if (head != NULL)
    {
        stack_reverse(&head, &(head->next));
    }
    printf("\nThe contents in stack after reversal\n");
    display(head);
    delete(&head);

    return 0;
}

void stack_reverse(struct node **head, struct node **head_next)
{
    struct node *temp;

    if (*head_next != NULL)
    {
         temp = (*head_next)->next;
        (*head_next)->next = (*head);
        *head = *head_next;
        *head_next = temp;
        stack_reverse(head, head_next);
    }
}

void display(struct node *head)
{
    if (head != NULL)
    {
        printf("%d  ", head->a);
        display(head->next);
    }
}

void generate(struct node **head)
{
    int num, i;
    struct node *temp;

    printf("Enter length of list: ");
    scanf("%d", &num);
    for (i = num; i > 0; i--)
    {
        temp = (struct node *)malloc(sizeof(struct node));
        temp->a = i;
        if (*head == NULL)
        {
            *head = temp;
            (*head)->next = NULL;
        }
        else
        {
            temp->next = *head;
            *head = temp;
        }
    }
}

void delete(struct node **head)
{
    struct node *temp;
    while (*head != NULL)
    {
        temp = *head;
        *head = (*head)->next;
        free(temp);
    }
}
```

Q16. John is a software developer working on a new project that requires him to write a C program to find the nth Fibonacci number using recursion. Help him to get the solution.

Solution 16:

```
#include <stdio.h>
int fibonacci(int num) {
    if (num == 0 || num == 1) {
        return num;
    } else {
        return fibonacci(num - 1) + fibonacci(num - 2);
    }
}
int main() {
    int num;
    scanf("%d", &num);
    printf("%d\n",fibonacci(num));
    return 0;
}
```

Q17. Create a union named "var" that includes an integer n, a float f, and a character c. Write a program to input data for each member of the union and display their values.

Solution 17:

```
#include<stdio.h>
union var {
    int i;
    float f;
    char c;
};
int main() {
    union var v;
    scanf("%d", &v.i);
    printf("%d\n", v.i);
    scanf("%f", &v.f);
    printf("%f\n", v.f);
    scanf(" %c", &v.c);
    printf("%c\n", v.c);
    return 0;
}
```

Q18 **Given an array of N non-negative integers arr[] representing an elevation map where the width of each bar is 1, compute how much water it is able to trap after raining.**

Solution:

```
 #include <stdio.h>

// Creating MACRO for finding the maximum number
#define max(x, y) (((x) > (y)) ? (x) : (y))

// Creating MACRO for finding the minimum number
#define min(x, y) (((x) < (y)) ? (x) : (y))

// Function to return the maximum
// water that can be stored
int maxWater(int arr[], int n)
{
// To store the maximum water
int res = 0;

// For every element of the array
for (int i = 0; i < n; i++) {

        // Find the maximum element on its left
        int left = arr[i];
        for (int j = 0; j < i; j++) {
            left = max(left, arr[j]);
        }

        // Find the maximum element on its left
        int right = arr[i];
        for (int j = i + 1; j < n; j++) {
            right = max(right, arr[j]);
        }

        // Update the result (maximum water)
        res = res + (min(left, right) - arr[i]);
    }

    // Return the maximum water
    return res;
    }
    int main()
    {
    int arr[] = { 0, 1, 0, 2, 1, 0, 1, 3, 2, 1, 2, 1 };
    int n = sizeof(arr) / sizeof(arr[0]);
    printf("%d", maxWater(arr, n));
    return 0;
    }
```

Q16. Define an enum named "Months" with January, February, March, April, May, June, July, August, September, October, November, December as its members. Write a program that takes an integer value from the user and prints the corresponding month.

Constraints:

0 < n<=12

Solution 16:

```
#include <stdio.h>

enum Months {
  January = 1, February, March, April, May, June, July, August, September, October, November, December
};

int main() {
   int monthNum;
   printf("Enter a month number (1-12): ");
   scanf("%d", &monthNum);

   switch (monthNum) {
      case January:
         printf("January");
         break;
      case February:
         printf("February");
         break;
      case March:
         printf("March");
         break;
      case April:
         printf("April");
         break;
      case May:
         printf("May");
         break;
      case June:
         printf("June");
         break;
      case July:
         printf("July");
         break;
      case August:
         printf("August");
         break;
      case September:
         printf("September");
         break;
      case October:
         printf("October");
         break;
      case November:
         printf("November");
         break;
      case December:
         printf("December");
         break;
      default:
         printf("Invalid month number");
         break;
   }
   return 0;
}
```

Q17. The atoi() function takes a string (which represents an integer) as an argument and returns its value. How to compute recursively.

Solution 17:

```
#include <stdio.h>
#include <string.h>
int myAtoiRecursive(char* str, int n)
{
int count = 0, check;
for (int i = 0; i <= strlen(str); ++i)
    {
            check = ( ( 'a' <= str[i] && str[i] <= 'z' ) ||
        ( 'A' <= str[i] && str[i] <= 'Z' ) )?1:0;
            if (check)
        {
            ++count;
        }
}
if (count != 0) {
        return 0;
}
if (n == 1)
        return *str - '0';
return (10 * myAtoiRecursive(str, n - 1) + str[n - 1]
                - '0');
}

// Driver Program
int main(void)
{
char str[10];
  gets(str);
int n = strlen(str);
printf("%d", myAtoiRecursive(str, n));
return 0;
}
```

Q18 **Imagine you are building a calculator application where the user can enter a number and perform various operations on it. To allow the user to perform calculations on the reverse of the entered number, you decide to write a function to find the reverse of the number using recursion. This function will take the user input as an argument, find its reverse using recursion, and return the reversed number for further calculations.**

Solution:

```
#include <stdio.h>

#include <math.h>



int rev(int, int);



int main()

{

  int num, result;

  int length = 0, temp;



  printf("Enter an integer number to reverse: ");

  scanf("%d", &num);

  temp = num;
```

```
    while (temp != 0)
    {
        length++;
        temp = temp / 10;
    }
    result = rev(num, length);
    printf("The reverse of %d is %d.\n", num, result);
    return 0;
}

int rev(int num, int len)
{
    if (len == 1)
    {
        return num;
    }
    else
    {
        return (((num % 10) * pow(10, len - 1)) + rev(num / 10, --len));
    }
}
```

Q16. Define a structure named Time to represent the time of day (hour, minute, and second). Write a program to read two times from the user, subtract them, and display the result on the console. see test cases for more clarification.

Solution 16:

```
#include <stdio.h>
struct Time {
    int hour;
    int minute;
    int second;
};
int main() {
    struct Time t1, t2, result;
    scanf("%d%d%d", &t1.hour, &t1.minute, &t1.second);
    scanf("%d%d%d", &t2.hour, &t2.minute, &t2.second);
    result.second = t1.second + t2.second;
    result.minute = t1.minute + t2.minute + result.second/60;
    result.hour = t1.hour + t2.hour + result.minute/60;
    result.second %= 60;
    result.minute %= 60;
    printf("%d:%d:%d\n", result.hour, result.minute, result.second);
    return 0;
}
```

Q17. Imagine you're a mathematician studying the properties of numbers. You come across a unique number and its square root, such that when the number is raised to the power of its square root, a fascinating result occurs. You have given a number and its square root. You have to determine what the result of this calculation would be?

As answers can be very large, print the result modulo 109 + 7.

Solution 17:

```
#include <stdio.h>
# define mod 1000000007
long long solve(int N,int R)
{
    if(R==0) return 1;
    long long temp = solve(N,R/2);
    temp=(temp*temp)%mod;
    if(R%2==1)
    {
        return (temp*N)%mod;
    }
    return temp;
}
int main()
{
    long long N,R;
    scanf("%ld%ld",&N,&R);
    long long ans =solve(N,R);
    printf("%ld",ans);
}
```

Q18 I**magine you are an astronaut preparing for a space mission. As part of the preparation, you need to know your weight on different planets to plan and manage your nutrition and exercise regimen during the mission. You use this program to input your current weight on Earth and obtain your weight on Mercury, Venus, Earth, and Mars planets.**

Solution:

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

// Define a struct for a planet
typedef struct {
    double mass;    // in kg
    double radius;  // in meters
} Planet;

// Define constants for each planet
const Planet MERCURY = {3.303e+23, 2.324397e8};
const Planet VENUS = {4.869e+24, 6.0518e6};
const Planet EARTH = {5.976e+24, 6.37814e6};
const Planet MARS = {6.421e+23, 3.3972e6};

// Define functions for calculating surface gravity and weight on each planet
double surfaceGravity(Planet p) {
    const double G = 6.67300E-11;
    return G * p.mass / (p.radius * p.radius);
}

double surfaceWeight(Planet p, double otherMass) {
    return otherMass * surfaceGravity(p);
}

int main() {
    // Prompt the user for their weight on Earth
    printf("Enter Your Weight : ");
    double earthWeight;
    scanf("%lf", &earthWeight);

    // Calculate the mass of the person based on their weight on Earth
    double mass = earthWeight / surfaceGravity(EARTH);

    // Print the person's weight on each planet
    printf("Your Weight on MERCURY is %f\n", surfaceWeight(MERCURY, mass));
    printf("Your Weight on VENUS is %f\n", surfaceWeight(VENUS, mass));
    printf("Your Weight on EARTH is %f\n", surfaceWeight(EARTH, mass));
    printf("Your Weight on MARS is %f\n", surfaceWeight(MARS, mass));

    return 0;
}
```