C (2071 shrawan) solution:

1. Categorise programming language on the basic of their uses and application. Among them which programming language is C programming?

Ans: Programming languages can be categorized based on their uses and applications into several different types:

1. General-Purpose Programming Languages:(c,c++,Python,Java,Ruby):The languages are versatile and can be used for a wide range of applications, from system programming to web development.
2. System Programming Languages(c,Assembly): These languages are used to develop low-level software like operating system,device drivers, and firmware.
3. Web development languages(Javascript,HTML,CSS,PHP,Ruby on Rails): These languages are used to create web applications, websites, and web services.
4. Database languages(SQL): These languages are used for managing and querying databases.
5. Scripting Languages(Python, Ruby,Perl,Javascript): These languages are often interpreted and used for automating tasks and writing short scripts.
6. Object-oriented Programming Langugaes: (c++,jaba,c#,Python):These languages revolve around the concept of objects and classes, making it easier to model real-world entities in code.
7. Embedded Programming languages(c) These languages are used for programming embedded systems and microcontrollers.

C programming falls into several categories:
#General-purpose programming languages: C is considered a general purpose programming language because it can be used for a wide range applications, from system programming to application development.
# system Programming language: C is commonly used for system programming due to its low level features and close relationship with hardware.
#Embedded Programming languages: C's efficiency and ability to directly manipulate hardware make it a popular choice for programming embedded systems and microcontrollers.

2.List and define different steps to solve the problem in computer systems.
Ans: Solving a problem in a computer system involves a systematic approach to identifying,analyzing, and implementing solutions
Here are the different steps involved in solving a problem in a computer system:
1.Problem definition: Clearly define the problem you're trying to solve. Understand the requirements and constraints.
2. Analysis: Breakdown the problem into smaller components and understand the relationship between them. Identify inputs, processes, and desired outputs.
3. Algorithm Design: design a algorithmic solution to the problem, determining the logic and sequence of steps needed to achieve the desired outcome.
4. Implementation: Write the actual code based on the pseudo code/flowchart.
5.Testing: Develop test cases that covers various scenarios and cases.

6. Monitor and verify: After implementing the solution, closely monitor the system to ensure that the problem does not reoccur. Verify th at the system’s performance and behavior have improved as expected.
7. Document the solution: Maintain detailed documentation of the problem, the chosen solution, the steps taken to implement its, and the outcomes. This documentation can be valuable for future reference.

8. Seek Feedback.
9. Reflect and learn

2.a. Differentiate between declaration and definition. Explain structure of C programming with an appropriate example:
Ans Declaration: A declaration introduces the name and type of a variable or function to the compiler. It tells the compiler about the existence and type of the variable or function without allocating memory or providing the actual implementation. Declaration are typically found in header files
Definition: A definition provides the actual implementation and memory allocation for a variable or function. It “creates” the entity in memory. Definitions are typically found in source code files.
For example:

```
// Declaration - Tells the compiler about the existence and type
extern int x;
void foo();

// Definition - Provides the actual implementation and memory allocation
int x = 10;
void foo() {
   // function implementation
}
```

Structure of C programming: C programming uses modular approach with functions and variables being organized in a structured manner. The structure of a c program consists of various elements, including directives, functions, variables, and statements.

```
// Preprocessor Directives - Used to include header files and perform text substitution
#include <stdio.h>

// Global Variables - Declared outside of functions, accessible throughout the program
int globalVar = 5;

// Function Declaration - Informs the compiler about the function's existence and signature
int add(int a, int b);

// Main Function - The entry point of the program
int main() {
   // Local Variable - Scoped to the main function
   int localVar = 10;

   // Function Call - Using the add function
   int result = add(globalVar, localVar);
```

```
    // Output the result
    printf("Result: %d\n", result);

    return 0;
}

// Function Definition - Provides the actual implementation
int add(int a, int b) {
    return a + b;
}
```

B. Write syntax,example and use of the following:
I. printf():
syntax:int printf(const char *format, ...);
Use: The printf() function is used to display the formatted output to the console. It takes a format string as first argument, which contains placeholders like ‘%d’,’%f’,’%s’,etc, that are replaced by the values of subsequent arguments.

ii.scanf():
syntax:int scanf(const char *format, ...);
Example:
scanf(“%d”,&age);
Use:The “scanf” function is used to read formatted input from
The user .
iii.getche():
Syntax: int getche();
Ch = getche();
Use: the getche() is used to read a single character from the standard input without waiting for the user to press the Enter key.
iv.getch():
Syntax:
Var_name = getch();
Example:
Ch = getch();
Use: to read a character without echoing it to the screen.

3. Write the difference between while and do……while loop and write the program “to find whether a year is leap or not”
Ans: In a ‘do….while’ loop, the body is executed before checking the condition. This guarantees that the loop body will execute at least once.
In a while loop, the condition is checked before executing the loop body. If the condition is false from the beginning, the loop body will not execute.

```
// program to find whether a year is leap year or not:
#include <stdio.h>
int main(){
        int n;
```

```
	printf("Enter year>> ");
	scanf("%d",&n);
	if (n%100 == 0){
		if(n%400==0){
			printf("leap year");
		}
		else{
			printf("not leap year");
		}
	}else if(n%4==0){
		printf("leap year");
	}else{
		printf("leap year");
	}
}
```

4.What is recursive function? How does it work? Find out sum of digit of number until the number becomes one digit number?
A recursive function is a function that calls itself directly or indirectly in order to solve a problem. Recursive function are commonly used in the programming to solve problems that can be broken down into smaller instances of the same problem.

A recursive function works by breaking down a complex problem into smaller.,more manageable subproblems of the same type. Each time the function is called,it works ona smaller instance of the original problem, and these smaller instances eventually lead to a base case where the recursion stops. The function then starts combining the results from teh base cases and the smaller instances to solve the original problem.

```
#include <stdio.h>

int sumOfDigits(int n) {
    // Base case: If n is a one-digit number, return n
    if (n < 10) {
        return n;
    }

    // Recursive case: Sum the digits and call the function recursively
    return n % 10 + sumOfDigits(n / 10);
}

int main() {
    int num;

    printf("Enter a number: ");
    scanf("%d", &num);

    int result = sumOfDigits(num);
    printf("Sum of digits of %d = %d\n", num, result);

    return 0;
}
```

5. Wap to read a string and rewrite its characters in alphabetic order.

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

int main() {
    char a[100];
    int i, j;
    char temp;

    printf("Enter a string: ");
    scanf("%s", a);

    printf("Original string: %s\n", a);

    for (i = 0; i < strlen(a) - 1; i++) {
        for (j = i + 1; j < strlen(a); j++) {
            if (a[i] > a[j]) {b
                temp = a[i];
                a[i] = a[j];
                a[j] = temp;
            }
        }
    }

    a[strlen(a)] = '\0'; // Add null character at the end

    printf("String in alphabetic order: %s\n", a);
     printf("%d",strlen(a));
    return 0;
}
```

B. A multinational company has hired 3 sales persons for marketing/selling its 3 different products in kathmandu. Each sale person sells each of these products. Write a program to read number of each product sold by all sale-persons. Calculate total sells of each item and the total sells of each sales-person. Use arrays:

Ans:
```
#include <stdio.h>

int main() {
    int sales[3][3]; // Array to store sales for each salesperson and each product
    int totalProductSales[3] = {0}; // Array to store total sales of each product
    int totalSalespersonSales[3] = {0}; // Array to store total sales of each salesperson

    // Reading sales data for each salesperson and each product
    for (int i = 0; i < 3; i++) {
        for (int j = 0; j < 3; j++) {
            printf("Enter number of product %d sold by salesperson %d: ", j + 1, i + 1);
            scanf("%d", &sales[i][j]);

            totalProductSales[j] += sales[i][j]; // Update total sales for each product
            totalSalespersonSales[i] += sales[i][j]; // Update total sales for each salesperson
        }
    }

    // Displaying total sales for each product
    printf("\nTotal sales for each product:\n");
    for (int j = 0; j < 3; j++) {
        printf("Product %d: %d\n", j + 1, totalProductSales[j]);
    }

    // Displaying total sales for each salesperson
    printf("\nTotal sales for each salesperson:\n");
    for (int i = 0; i < 3; i++) {
        printf("Salesperson %d: %d\n", i + 1, totalSalespersonSales[i]);
    }

    return 0;
}
```

6.Explain about array with structures along with programming example:

An array of structures is a data structure that allows you to create a collection of elements, where each element is of a structured data type.In other words, you can create an array in which each element is a structure.This can be useful when you want to ground related data together and store multiple instances of that grouped data in an organized manner.

Example of defining a structure to represent a book:

```
 Struct book{
    Char title[100];
Char author[50];
Int year;
};
#include <stdio.h>

struct Book {
    char title[100];
    char author[50];
    int year;
};
```

```
int main() {
    struct Book library[3]; // Array of structures

    // Input data for each book
    for (int i = 0; i < 3; i++) {
        printf("Enter details for book %d:\n", i + 1);
        printf("Title: ");
        scanf("%s", library[i].title); // Note: No & before library[i].title
        printf("Author: ");
        scanf("%s", library[i].author);
        printf("Year: ");
        scanf("%d", &library[i].year);
    }

    // Display data for each book
    printf("\nLibrary:\n");
    for (int i = 0; i < 3; i++) {
        printf("Book %d\n", i + 1);
        printf("Title: %s\n", library[i].title);
        printf("Author: %s\n", library[i].author);
        printf("Year: %d\n", library[i].year);
    }

    return 0;
}
```

6.b. Write the program to understand how the structure members are sent to a function.

Ans:
```
#include <stdio.h>

struct Point {
    int x;
    int y;
};

// Function to print the coordinates of a point
void printPointCoordinates(int x, int y) {
    printf("Coordinates: (%d, %d)\n", x, y);
}

// Function to print the coordinates of a point using a structure
void printPoint(struct Point p) {
    printf("Coordinates: (%d, %d)\n", p.x, p.y);
}

int main() {
    struct Point myPoint;
    printf("Enter x coordinate: ");
    scanf("%d", &myPoint.x);

    printf("Enter y coordinate: ");
    scanf("%d", &myPoint.y);

    // Passing individual members to the function
    printPointCoordinates(myPoint.x, myPoint.y);

    // Passing the whole structure to the function
    printPoint(myPoint);

    return 0;
}
```

7. Write down advantage of pointer. Write a program using pointer to swap the value of two variables where the swapping operation is performed in separate function.
Ans: Advantage of pointers:
I. Memory efficiency: Pointer allows you to work directly with memory addresses. Which can be more-efficient than storing and passing around copies of data.
Ii. Dynamic memory allocation: Pointers are essential for managing dynamically allocated memory, such as memory allocated using functions like ‘malloc()’ or ‘calloc()’.
Iii. Efficient Parameter passing: Passing a pointer to a function can be more efficient than passing large structures or arrays by values.
Iv. String manipulation: Pointers are used extensively in string manipulation and handling.

```
#include <stdio.h>

void swap(int *a, int *b) {
    int temp = *a;
    *a = *b;
    *b = temp;
}

int main() {
    int x, y;

    printf("Enter the value of x: ");
    scanf("%d", &x);

    printf("Enter the value of y: ");
    scanf("%d", &y);

    printf("Before swapping: x = %d, y = %d\n", x, y);

    // Pass pointers to swap function
    swap(&x, &y);
    printf("After swapping: x = %d, y = %d\n", x, y);

    return 0;
}
```

8. Write a c program to store employee details in a text file. Read data from the text file, sort them in ascending order of salary and store the sorted record to a binary file . Display the details and rank of employee given by the user.
Ans:

```
// Write a c program to store employee details in a text file. Read data from the text file, sort them in ascending order of salary and store
//the sorted record to a binary file . Display the details and rank of employee
//given by the user.
#include <stdio.h>
struct employee{
        char name[100];
        int sal;
};

int main (){
        struct employee emp[3]={
        {"a",5500},
        {"b",4500},
        {"c",6500}
        };
// stroing employee details in text file:
int i;
FILE *fp = fopen("employeedetails.txt","w");
if (fp==NULL){
        printf("File doesn't exist.");
}

for (i=0;i<3;i++){
  fprintf(fp,"%s %d",emp[i].name,emp[i].sal);
}

fclose(fp);
// read data from text file and sort them in ascending order:
fp = fopen("employeedetails.txt","r");
if (fp==NULL){
        printf("File doesn't exist");
}
struct employee em[3];
for (i=0;i<3;i++){
        fscanf(fp,"%s %d",em[i].name,&em[i].sal);
        printf("\n%s %d\n",em[i].name,em[i].sal);
}
// sort the employess on  the basis of salary:
int j; struct employee temp;
for (i=0;i<2;i++){
        for(j=i+1;j<3;j++){
                if(em[i].sal > em[j].sal){
                        temp = em[i];
                        em[i]=em[j];
                        em[j]=temp;
                }
        }
}
// stroing and displaying sorted employee details:
fp = fopen("employeedetails.bin","wb");
for (i=0;i<3;i++){
        printf("\n%s %d",em[i].name,em[i].sal);
        fprintf(fp,"%s %d",em[i].name,em[i].sal);
}
}
```

WHAT DO YOU MEAN BY formamtted and unformatted input/ output statements in Fortan and also give some suitable example with concept of formatted I/

Ans: In fortran, formatted and unformatted input/output (i/o) statement refer to the methods of reading and writing data to and from files, terminals, or other devices.
Formatted I/O:
Formatted I/O involves using specific format descriptors to control the appearance of the data when it is read from or written to a file or device.
Format descriptors define the field width, decimal places, alignment, and other formatting options for data. Formatted I/O is more human-readable and provides control over the appearance of data, but it can slower compared to unformat ted I?O due to the additional processing required for formatting.

```
program FormattedOutput
  implicit none
  integer :: i, j
  real :: pi
  character(10) :: name

  i = 42
  j = -17
  pi = 3.14159
  name = "John"

  ! Writing data to a file with formatted output
  open(unit=10, file='formatted_output.txt', status='replace')
  write(10, '(A, I5, A, I5, A, F8.3, A, A)') "i = ", i, ", j = ", j, ", pi = ", pi, ", name = ", name
  close(10)
```

```
end program FormattedOutput
```

Unformatted I/O:d
nformatted I/O reads or writes raw binary data without any formatting or additional characters. It is generally faster than formatted I/O because it doesn't involve the parsing and formatting of data. However, the resulting file may not be as human-readable as with formatted I/O.

```
program UnformattedOutput
    implicit none
    integer :: i, j
    real :: pi
    character(10) :: name

    i = 42
    j = -17
    pi = 3.14159
    name = "John"

    ! Writing data to a file with unformatted output
    open(unit=20, file='unformatted_output.dat', status='replace', form='unformatted')
    write(20) i, j, pi, name
    close(20)

end program UnformattedOutput
```

9.b. Write a program to convert binary number to a decimal number using fortran programming language.

```
program BinaryToDecimal
    implicit none
    character(20) :: binary
    integer :: decimal, i, length

    ! Read the binary number as a string
    write(*,*) "Enter a binary number: "
    read(*,*) binary

    ! Calculate the length of the binary string
    length = len_trim(binary)

    ! Convert binary to decimal
    decimal = 0
    do i = 1, length
```

```
        decimal = decimal + (int(binary(i:i)) * 2**(length - i))
    end do

    ! Display the result
    write(*,*) "Decimal equivalent: ", decimal

end program BinaryToDecimal
```