Programming For Problem Solving

Section-C

1. Write a program to reverse the digit number and sum of the digits ?

```c
#include <stdio.h>

int main() {
    int number, reversedNumber = 0, digitSum = 0;

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

    while (number > 0) {
        int digit = number % 10;
        reversedNumber = reversedNumber * 10 + digit;
        digitSum += digit;
        number /= 10;
    }

    printf("Reversed number: %d\n", reversedNumber);
    printf("Sum of digits: %d\n", digitSum);

    return 0;
}
```

2. What are the user defined data types in C ? explain with example.

1. **Structures**: Structures allow you to group different variables of different data types into a single entity. You can define a structure using the `struct` keyword. Here's an example:

```c
#include <stdio.h>

// Define a structure representing a point in 2D space
struct Point {
    int x;
    int y;
};

int main() {
    // Declare a variable of type 'struct Point'
    struct Point p;

    // Assign values to the structure members
    p.x = 10;
    p.y = 5;

    // Access and print the structure members
    printf("Coordinates: (%d, %d)\n", p.x, p.y);

    return 0;
}
```

2. **Enumerations**: Enumerations allow you to define a custom set of named values. Each named value represents a constant integer. You can define an enumeration using the `enum` keyword. Here's an example:

```c
#include <stdio.h>

// Define an enumeration representing days of the week
enum Weekday {
    Monday,
    Tuesday,
    Wednesday,
    Thursday,
    Friday,
    Saturday,
    Sunday
};

int main() {
    // Declare a variable of type 'enum Weekday'
    enum Weekday today;

    // Assign a value from the enumeration
    today = Tuesday;

    // Print the value
    printf("Today is %d\n", today);

    return 0;
}
```

3. write program using function to find the all types of roots of quadratic equation.
4. write a program to find some of two matrices.

```c
#include<stdio.h>
#include<conio.h>
int main()
{
    int a[2][2],b[2][2],c[2][2],i,j;
    printf("ENTER MATRIX FOR A(2x2) :\n");
    for ( i = 0; i <2; i++)
    {
     for ( j = 0; j <2; j++)
     {
        scanf("%d",&a[i][j]);
     }

    }
    printf("ENTER MATRIX FOR B(2x2) :\n");
    for ( i = 0; i <2; i++)
    {
     for ( j = 0; j <2; j++)
     {
        scanf("%d",&b[i][j]);
     }

    }
    printf(" SUBTRACTION OF MATRIX : \n");
    for ( i = 0; i <2; i++)
    {
     for ( j = 0; j <2; j++)
     {
        c[i][j]=a[i][j]-b[i][j];
        printf("%d\t",c[i][j]);
     }
     printf("\n");
    }
}
```

5. What are the various types of software used in computer? Explain each of them with example.

1. **Operating System (OS)**: **An operating system is the core software that manages computer hardware and provides essential services for other software applications. It acts as an interface between the user and the computer hardware. Examples of operating systems include Windows, macOS, Linux, and Android.**

2. **Application Software: Application software is designed to perform specific tasks or functions for the user. It helps users accomplish various activities on their computers, such as word processing, web browsing, graphic design, and video editing. Examples of application software include Microsoft Word, Google Chrome, Adobe Photoshop, and VLC Media Player.**

3. **Utility Software: Utility software provides additional functionalities to enhance the performance, security, and management of the computer system. It includes tools for system maintenance, data backup, antivirus protection, disk defragmentation, and file compression. Examples of utility software include Norton Antivirus, CCleaner, WinZip, and Disk Cleanup.**

4. **Programming Software: Programming software provides tools and environments for developers to create, test, and debug computer programs. It includes text editors, integrated development environments (IDEs), compilers, debuggers, and version control systems. Examples of programming software include Visual Studio, Eclipse, Xcode, and Git.**

5. **Device Drivers: Device drivers are software programs that enable communication between the operating system and specific hardware devices. They allow the operating system to control and utilize hardware components such as printers, scanners, sound cards, and network adapters. Device drivers are often provided by the hardware manufacturers or included with the operating system.**

6. **Database Management Systems (DBMS): DBMS software is used to manage large collections of data and provide efficient storage, retrieval, and manipulation of that data. It allows users to create, organize, and query databases. Examples of DBMS software include Oracle Database, MySQL, Microsoft SQL Server, and MongoDB.**

7. **Enterprise Resource Planning (ERP): ERP software integrates various business processes and functions within an organization, such as finance, human resources, inventory management, and customer relationship management. It helps streamline operations and improve overall efficiency. Examples of ERP software include SAP, Oracle ERP, Microsoft Dynamics, and NetSuite.**

8. **Graphics and Design Software**: **Graphics and design software are used for creating and manipulating visual content, including images, illustrations, animations, and 3D models. They are widely used in graphic design, architecture, animation, and multimedia industries. Examples of graphics and design software include Adobe Photoshop, Illustrator, AutoCAD, and Blender.**

6. What is the specification of a Latest PC? Explain the functionality of each of the component.

1. **Processor (CPU):** The processor is the central processing unit of a computer that performs most of the computational tasks. It executes instructions, performs calculations, and manages the flow of data. Modern processors, such as Intel Core i7 or AMD Ryzen series, have multiple cores and high clock speeds, allowing for efficient multitasking and faster processing of applications.

2. **Memory (RAM):** Random Access Memory (RAM) is the temporary storage space used by the computer to hold data that is actively being used. It provides quick access to data and instructions for the processor. Higher RAM capacity, such as 8GB, 16GB, or even 32GB, allows for smooth multitasking and faster data processing.
3. **Storage (Hard Drive/SSD)**: Storage devices are used to store data, files, and the operating system. Hard Disk Drives (HDD) provide large storage capacities at a lower cost, while Solid-State Drives (SSD) offer faster read/write speeds and improved performance. Many modern PCs come with a combination of HDD and SSD, where the SSD is used as the primary drive for faster system responsiveness.
4. **Graphics Card (GPU):** A dedicated Graphics Processing Unit (GPU) is responsible for rendering and displaying images, videos, and graphical content on the monitor. It offloads the graphics-related tasks from the CPU, resulting in smoother graphics performance, gaming, and multimedia experiences. High-end GPUs, such as NVIDIA GeForce RTX or AMD Radeon RX series, are capable of handling demanding graphics-intensive applications and gaming.
5. **Motherboard:** The motherboard serves as the main circuit board that connects and integrates various components of the computer. It provides the communication pathway between the CPU, RAM, storage, and other peripherals. The motherboard also houses essential connectors, slots, and ports for expansion cards, USB devices, network connections, and audio devices.
6. **Power Supply Unit (PSU):** The PSU is responsible for converting and supplying the necessary power to the different components of the computer. It ensures stable and consistent power delivery to avoid hardware failures. The PSU's capacity is determined by the total power requirements of the components and should be sufficient to handle the overall system's power needs.
7. **Monitor:** The monitor is the display output device that allows users to visualize and interact with the computer's graphical interface. Modern monitors come in various sizes, resolutions (e.g., Full HD, 4K), and display technologies (e.g., LCD, LED, OLED) to provide high-quality visuals and enhance the overall user experience.
8. **Input Devices:** Input devices include the keyboard and mouse, which enable users to interact with the computer and provide input for various applications. Additionally, touchscreens, stylus pens, and other specialized input devices are becoming more prevalent, particularly in tablets, 2-in-1 laptops, and touch-enabled PCs.

7. **What are the various Control Structures available in a programming Language? Explain with example.**

1. **Conditional Statements**:

- **if-else**: **It allows the program to execute different blocks of code based on a condition. If the condition is true, the code inside the if block is executed; otherwise, the code inside the else block is executed.**

```
int age = 18;
if (age >= 18) {
    printf("You are an adult.\n");
} else {
    printf("You are not an adult.\n");
}
```

- **switch-case:** It allows the program to perform different actions based on different possible values of an expression.

```c
char grade = 'B';
switch (grade) {
    case 'A':
        printf("Excellent!\n");
        break;
    case 'B':
        printf("Good!\n");
        break;
    case 'C':
        printf("Average!\n");
        break;
    default:
        printf("Invalid grade!\n");
}
```

2. **Loops**:

- **for loop**: It allows the program to repeat a block of code for a specified number of times.

```c
int i;
for (i = 1; i <= 5; i++) {
    printf("%d ", i);
}
printf("\n");
```

- **while loop**: It allows the program to repeat a block of code as long as a given condition is true.

```c
int i = 1;
while (i <= 5) {
    printf("%d ", i);
    i++;
}
printf("\n");
```

- **do-while loop:** It is similar to the while loop, but the condition is checked after executing the block of code, ensuring the code is executed at least once.

```c
int i = 1;
do {
    printf("%d ", i);
    i++;
} while (i <= 5);
printf("\n");
```

3. **Control Statements:**

1. **break:** It terminates the current loop or switch-case statement and transfers control to the next statement outside the loop or switch-case.

```c
int i;
for (i = 1; i <= 5; i++) {
    if (i == 3) {
        break;
    }
    printf("%d ", i);
}
printf("\n");
```

- **continue**: **It skips the remaining code within a loop and proceeds to the next iteration.**

```c
int i;
for (i = 1; i <= 5; i++) {
    if (i == 3) {
        continue;
    }
    printf("%d ", i);
}
printf("\n");
```

8. Write a program to find the difference of two matrix using functions.

```c
#include <stdio.h>

// Function to subtract two matrices
void subtractMatrices(int matrix1[][3], int matrix2[][3], int result[][3]) {
    int i, j;

    for (i = 0; i < 3; i++) {
        for (j = 0; j < 3; j++) {
            result[i][j] = matrix1[i][j] - matrix2[i][j];
        }
    }
}

int main() {
    int matrix1[3][3] = {
        {1, 2, 3},
        {4, 5, 6},
        {7, 8, 9}
    };

    int matrix2[3][3] = {
        {9, 8, 7},
        {6, 5, 4},
        {3, 2, 1}
    };

    int result[3][3];
    int i, j;

    // Subtract matrices
    for (i = 0; i < 3; i++) {
        for (j = 0; j < 3; j++) {
            result[i][j] = matrix1[i][j] - matrix2[i][j];
        }
    }

    // Display result
    printf("Difference of the matrices:\n");
    for (i = 0; i < 3; i++) {
        for (j = 0; j < 3; j++) {
            printf("%d ", result[i][j]);
        }
        printf("\n");
    }
return 0;
}
```

9. What is an array ? write a program in 'C' to subtract two Matrices .

Ans: Array: An array is a collection of items of same data type stored at contiguous memory locations.

```c
#include<stdio.h>
#include<conio.h>
int main()
{
int a[2][2],b[2][2],c[2][2],i,j;
printf("ENTER MATRIX FOR A(2x2) :\n");
for ( i = 0; i <2; i++)
{
for ( j = 0; j <2; j++)
{
scanf("%d",&a[i][j]);
}
}
printf("ENTER MATRIX FOR B(2x2) :\n");
for ( i = 0; i <2; i++)
{
for ( j = 0; j <2; j++)
{
scanf("%d",&b[i][j]);
}
}
printf(" SUBTRACTION OF MATRIX : \n");
for ( i = 0; i <2; i++)
{
for ( j = 0; j <2; j++)
{
c[i][j]=a[i][j]-b[i][j];
printf("%d\t",c[i][j]);
}
printf("\n");
}
}
```

10. what are the various file opening statement available in C ?

11. **"r"** - Read Mode:
    a. Opens a file for reading.
    b. Returns NULL if the file does not exist.
    c. Example: `FILE *file = fopen("example.txt", "r");`
12. **"w"** - Write Mode:
    a. Opens a file for writing.
    b. If the file exists, its contents are truncated (deleted).

c. If the file does not exist, a new file is created.
d. Example: `FILE *file = fopen("example.txt", "w");`

13. **"a"** - Append Mode:
    a. Opens a file for appending data.
    b. If the file exists, new data is written at the end of the file.
    c. If the file does not exist, a new file is created.
    d. Example: `FILE *file = fopen("example.txt", "a");`
14. **"r+"** - Read/Write Mode:
    a. Opens a file for both reading and writing.
    b. The file must exist.
    c. Example: `FILE *file = fopen("example.txt", "r+");`
15. **"w+"** - Read/Write Mode (overwrite):
    a. Opens a file for both reading and writing.
    b. If the file exists, its contents are truncated (deleted).
    c. If the file does not exist, a new file is created.
    d. Example: `FILE *file = fopen("example.txt", "w+");`
16. **"a+"** - Read/Append Mode:
    a. Opens a file for both reading and appending data.
    b. If the file exists, new data is written at the end of the file.
    c. If the file does not exist, a new file is created.
    d. Example: `FILE *file = fopen("example.txt", "a+");`

11. what are the methods through which the parameters are passed to a function in C language ? How do they make a difference?

Ans : In C, parameters can be passed to a function using two methods: pass by value and pass by reference

Pass by Value:

- In pass by value, a copy of the value of the parameter is passed to the function.
- The function works with the copy of the parameter, not the original variable.
- Changes made to the parameter inside the function do not affect the original variable.
- Example:

```c
void increment(int num) {
    num++;
}

int main() {
    int x = 5;
    increment(x);
    printf("%d\n", x);  // Output: 5 (no change)
    return 0;
}
```

Pass by Reference:

- In pass by reference, the address (or reference) of the parameter is passed to the function.
- The function can directly access and modify the original variable using the reference.
- Changes made to the parameter inside the function are reflected in the original variable.
- Example:

```c
void increment(int* num) {
    (*num)++;
}

int main() {
    int x = 5;
    increment(&x);
    printf("%d\n", x);  // Output: 6 (changed)
    return 0;
}
```

12. Using shorting method of your choice ,write a program to sort 8 numbers.

```c

#include <stdio.h>

int main() {
    int arr[8] = {5, 2, 9, 1, 8, 6, 3, 7};
    int temp;
    for (int i = 0; i < 8 - 1; i++) {
        for (int j = 0; j < 8 - i - 1; j++) {
            if (arr[j] > arr[j + 1]) {
                temp = arr[j];
                arr[j] = arr[j + 1];
                arr[j + 1] = temp;
            }
        }
    }
    printf("Sorted array in ascending order: ");
    for (int i = 0; i < 8; i++) {
        printf("%d ", arr[i]);
    }
    return 0;
}
```

13. Explain the following terms in c language with example.

a) Static variable

b) External variable

c) Automatic variable

Ans: a) Static variable : In C, a static variable is a variable that retains its value even after the scope in which it is defined has ended. It is initialized only once, and its value persists throughout the program's execution. Static variables are declared using the `static` keyword.

```c
#include <stdio.h>

void increment() {
static int value = 0;  // Static variable

value++;
printf("value: %d\n", value);
}

int main() {
increment();  // Output: value: 1
increment();  // Output: value: 2
increment();  // Output: value: 3

return 0;
}
```

b) External variable : An external variable is a variable that is declared outside of any function, typically at the top of the file (global scope). It can be accessed and modified by any function within the file without the need for passing it as a parameter. External variables are declared using the `extern` keyword.

```c
#include <stdio.h>

extern int globalVar;  // External variable declaration

void printGlobalVar() {
    printf("Global Variable: %d\n", globalVar);
}

int globalVar = 10;  // External variable definition

int main() {
    printGlobalVar();  // Output: Global Variable: 10

    return 0;
}
```

c) Automatic Variable:

An automatic variable, also known as a local variable, is a variable that is defined within a block or function. It is created when the block or function is entered and destroyed when it is exited. Automatic variables have automatic storage duration, meaning they are allocated and deallocated automatically .

```c
#include <stdio.h>

void printMessage() {
    int count = 0;  // Automatic variable

    count++;
    printf("Count: %d\n", count);
}

int main() {
    printMessage();  // Output: Count: 1
    printMessage();  // Output: Count: 1

    return 0;
}
```