Structured Programming

CSE 103

Promod Chandra Das

231002005

Multiple Choice Questions

50. C programming language was developed by
A) Dennis Ritchie

69. Which of the following is an example of compounded assignment statement?
B) a += 5

102. C supports how many basic looping constructs

C) 4

81. Operators have precedence. A Precedence determines which operator is
C) evaluated first

121. C allows arrays of greater than two dimensions, who will determined this
B) compiler

115. A compound statement is a group of statements included between a pair of
B) curly braces

61. Character constants should be enclosed between

A) Single quotes

74. Operators have hierarchy. It is used to know which operator
B) is used first

80. Which operator has the highest priority?
A) ++

59. Which escape character can be used to begin a
new line in C?
D) \n

64. The maximum length of a variable in C is
C) 32

Very short questions

92.If x = 8.8,y = 3.5,z = -5.2, then determine value of following expression

a) 2 * y + 3 * (x - z) b) 2 * x / (3 * y)

Ans: -

Let's substitute the given values of x, y, and z into the expressions:

a) $2 \cdot y + 3 \cdot (x - z)$ $2 \cdot y + 3 \cdot (x - z)$ Substitute y = 3.5, x = 8.8, and z = -5.2:

$2 \cdot 3.5 + 3 \cdot (8.8 - (- 5.2))$

$7 + 3 \cdot (8.8 + 5.2)$

$7 + 3 \cdot 14$

7 + 42 7

The value is 49.

b) $2 \cdot x / (3 \cdot y)$

Substitute x = 8.8 and y = 3.5:

$2 \cdot 8.8 / (3 \cdot 3.5)$

17.6 / 10.5

The value is approximately 1.676.

Therefore: a) $2 \cdot y + 3 \cdot (x - y) = 49$

c) $2 \cdot x / (3 \cdot y) \approx 1.676$ $2 \cdot x/(3 \cdot y) \approx 1.676$

45. What is the use of strcmp() function?

Ans:

The strcmp() function in C and C++ is used to compare two strings. It stands for "string compare." This function takes two strings as arguments and returns an integer value that indicates the relationship between the strings.

2. What is a constant?

Ans:-

A constant is a fixed value that cannot be changed during the execution of a program.

67. Explain putchar()?

Ans:- putchar() is a standard C library function used to write a single character to the standard output (usually the console). It takes an integer (representing a character) as its argument and returns the character written.

36. What is a Logical Error?

Ans:- A logical error is a mistake in the algorithm or logic of a program that leads to incorrect results or unexpected behavior. Unlike syntax errors, logical errors do not cause the program to crash but may produce inaccurate or unintended outcomes. Debugging logical errors typically involves analyzing the program's design and flow to identify and correct the flawed logic.

81. Write a rule for declaring numeric constant.

Ans:-

Rule for declaring a numeric constant: Use the const keyword followed by the data type and a meaningful name, then assign the constant value.

10. How long is a word?

Ans:- The length of a word can vary depending on context and language. In computing and programming, a "word" often refers to the basic unit of data a computer's CPU can process in one operation. This length is typically 32 or 64 bits (4 or 8 bytes) on modern computers. In natural language, the length of a word varies, but it is generally a sequence of characters separated by spaces.

49. What is a recursive function?

Ans:- A recursive function is a function that calls itself, either directly or indirectly, in order to solve a problem or perform a task.

74. write a syntax of while loop?

Ans:-

```
while (condition) {
   // code to be executed while the condition is true
}
```

29. Give syntax of a simple if statement.

Ans:-

```
if (condition) {    // code to be executed if the condition is true }
```

102. Write a program to find the largest between two numbers?

Ans:-

```
#include <stdio.h>

int main() {
   // Declare variables
   int num1, num2;

   // Input two numbers
   printf("Enter the first number: ");
   scanf("%d", &num1);

   printf("Enter the second number: ");
   scanf("%d", &num2);

   // Determine the largest number
   if (num1 > num2) {
      printf("The largest number is: %d\n", num1);
   } else {
      printf("The largest number is: %d\n", num2);
   }

   return 0;
}
```

Short question

131. What is the automatic variable and what is the
use of it.

Ans:- Automatic variables in programming, especially in languages like C and C++, are local variables that are created and initialized every time a function is called, and they are destroyed when the function exits. These variables are typically declared within the body of a function and are used for temporary storage. Use of automatic variables: Temporary Storage , Scope Limited to Function, Efficient Memory Usage

9. Differentiate between relational and logical operators used in C?

Ans:

Relational Operators: Relational operators are used to compare two values.
They return a Boolean result (true or false) based on the comparison.
Common relational operators include == (equal to), != (not equal to), < (less than), > (greater than), <= (less than or equal to), and >= (greater than or equal to).

Logical Operators: Logical operators are used to perform logical operations on Boolean values (true or false). They operate on Boolean expressions and return a Boolean result. Common logical operators include && (logical AND), || (logical OR), and ! (logical NOT).

69. Explain nested structure.

Ans:-

A nested structure in programming refers to a structure (or record) that is defined within another structure. This means that the inner structure is a member of the outer structure. This concept is commonly used to organize and represent more complex data with a hierarchical or nested structure.

41. Explain how to access a value using pointer?give
a suitable example.

Ans:-

Ans:- To access a value using a pointer in C, you need to use the dereference operator (*). The dereference operator is applied to a pointer to obtain the value it points to.

int num = 42; declares an integer variable num and initializes it with the value 42.

int *ptr = # declares a pointer ptr to an integer and assigns the address of the variable num to it.

printf("Value of num: %d¥n", *ptr); prints the value of num by dereferencing the pointer using *ptr.

99. Explain the following bitwise operators:

i)Bitwise AND ii) Bitwise OR iii) BitwiseXOR

iv) Bitwise Left Shift v) Bitwise Right Shift

Ans:- i) Bitwise AND (&): Performs a bitwise AND operation between corresponding bits of two operands.

If both bits are 1, the result bit is set to 1; otherwise, it's set to 0.

ii) Bitwise OR (|): Performs a bitwise OR operation between corresponding bits of two operands.

If at least one of the bits is 1, the result bit is set to 1; otherwise, it's set to 0.

iii) Bitwise XOR (^): Performs a bitwise exclusive OR (XOR) operation between corresponding bits of two operands.

If the bits are different, the result bit is set to 1; if the bits are the same, the result bit is set to 0.

iv) Bitwise Left Shift (<<): Shifts the bits of the left operand to the left by a specified number of positions. Filled with zeros on the right. For example, a << 2 shifts the bits of a two positions to the left.

v) Bitwise Right Shift (>>):

55. Write a short note on file handling in C

Ans:-

Opening a File: fopen() function is used to open a file. It takes two arguments: the name of the file and the mode in which the file is to be opened (read, write, append, etc.).
Reading from a File: fscanf() or fgets() functions are used to read data from a file. Writing to a File: fprintf() or fputs() functions are used to write data to a file.
Writing to a File: fprintf() or fputs() functions are used to write data to a file.
Closing a File: fclose() function is used to close a file after performing operations.

88. What is type casting? Explain it with suitable example.

Ans:-

Type casting, also known as type conversion, is the process of converting a value from one data type to another. This is necessary when you want to perform operations or assignments involving variables of different data types. In C, type casting can be explicit or implicit.

37. Explain what is pointer?expalin with suitable example

Ans:- A pointer in C is a variable that stores the memory address of another variable. It allows you to indirectly manipulate the value of a variable by referencing its address in the computer's memory.

```
#include <stdio.h>

int main() {
   int number = 42;  // Declare an integer variable
   int *pointer;     // Declare a pointer to an integer

   pointer = &number; // Assign the address of 'number' to 'pointer'

   printf("Value of number: %d\n", number);
   printf("Address of number: %p\n", (void*)&number); // Using %p to print the address
   printf("Value stored at the address pointed by pointer: %d\n", *pointer);

   // Changing the value through the pointer
   *pointer = 99;

   printf("New value of number: %d\n", number);

   return 0;
}
```

79. Write a rule for declaring character constant

Ans:- To declare a character constant in C, use single quotes around the character. Here's the rule:
const char MY_CHAR = 'A';

146. Summerise the purpose of string.h function

Ans:- String Copy (strcpy): Copies the content of one string to another. String Concatenation (strcat): Concatenates (appends) one string to the end of another.
String Length (strlen): Returns the length (number of characters) in a string. String Comparison (strcmp): Compares two strings lexicographically.
String Search (strstr): Finds the first occurrence of a substring within a string. String Tokenization (strtok): Breaks a string into a series of tokens based on a specified delimiter. String Comparison (Case-insensitive)
(strcasecmp): Compares two strings without considering case. String Copy with Limited Characters
(strncpy): Copies a specified number of characters from one string to another. String Concatenation with Limited Characters
(strncat): Concatenates a specified number of characters from one string to another. String Comparison with Limited Characters (strncmp): Compares a specified number of characters in two strings.

14. Explain any two bitwise operator with suitable
example.
Ans:
Bitwise AND (&): • The bitwise AND operator (&) performs a bitwise AND operation between corresponding bits of two operands. • If both bits are 1, the result bit is set to 1; otherwise, it's set to 0.
Example : unsigned int a = 5; // Binary: 0101
unsigned int b = 3; // Binary: 0011
unsigned int result = a & b; // Binary result: 0001 (decimal: 1)

Bitwise OR (|): The bitwise OR operator (|) performs a bitwise OR operation between corresponding bits of two operands. If at least one of the bits is 1, the result bit is set to 1; otherwise, it's set to 0.

Example :

```
unsigned int a = 5;  // Binary: 0101
unsigned int b = 3;  // Binary: 0011
unsigned int result = a | b;  // Binary result: 0111 (decimal: 7)
```

70. Explain array of structure.

Ans:-

An array of structures in C is a data structure that allows you to store multiple instances of a structure type in a contiguous block of memory. Each element of the array is a structure, and you can access individual members of each structure using array indexing.

initialize and access each element of the array like any other array, setting and retrieving values for each member of the structure. Arrays of structures are useful when you need to manage a collection of records, where each record has multiple attributes.

105. What are the salient features of standard input

Ans:- he standard input in computer programming refers to the default input source from which a program reads data. In C, standard input is typically associated with the keyboard, and it can be accessed using functions like scanf and getchar.

the program uses scanf to read an integer from standard input, which is assumed to be the keyboard.

66. Explain dynamic memory allocation in brief.

Ans:- Dynamic memory allocation refers to the process of allocating memory for variables at runtime, as opposed to compile time. In languages like C and C++, dynamic memory allocation is achieved using functions like malloc, calloc, realloc, and free.

Dynamic memory allocation is particularly useful when the size of the data needed is not known at compile time or when data needs to be allocated and deallocated dynamically during program execution. However, it requires careful management to avoid memory leaks and fragmentation.

25. Explain nested for loop with an example

Ans:- A nested for loop is a loop inside another loop. This allows for repeated execution of a block of code within another block of code. Each iteration of the outer loop triggers the inner loop to complete its entire cycle.

```
#include <stdio.h>

int main() {
    // Outer loop for rows
    for (int i = 1; i <= 5; ++i) {
        // Inner loop for columns
        for (int j = 1; j <= 5; ++j) {
            printf("%4d", i * j);  // Adjusting spacing for a neat table
        }
        // Move to the next line after each row
        printf("¥n");
    }

    return 0;
}
```

150. What is mean by member or field of structure

Ans:- In the context of structures in programming, a member or field refers to an individual data item within the structure. A structure is a user-defined data type that allows you to group together variables of different types under a single name. Each variable within the structure is referred to as a member or field.

Programming Segment

2. Write a C Program to sort an array in ascending Order

Ans:-

```
#include <stdio.h>

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

void bubbleSort(int arr[], int n) {
    for (int i = 0; i < n - 1; ++i) {
        for (int j = 0; j < n - i - 1; ++j) {
            if (arr[j] > arr[j + 1]) {
                // Swap if the element found is greater than the next element
                swap(&arr[j], &arr[j + 1]);
            }
        }
    }
}

void printArray(int arr[], int size) {
    for (int i = 0; i < size; i++) {
        printf("%d ", arr[i]);
    }
    printf("¥n");
}

int main() {
    int arr[] = {64, 34, 25, 12, 22, 11, 90};
    int n = sizeof(arr) / sizeof(arr[0]);
```

```
    printf("Original array: ");
    printArray(arr, n);

    // Sort the array
    bubbleSort(arr, n);

    printf("Sorted array in ascending order: ");
    printArray(arr, n);

    return 0;
}
```

11. Write a C Program to print equivalent hex number of given decimal number

Ans:-

```
#include <stdio.h>

int main() {
    // Declare variables
    int decimalNumber;

    // Input: Get the decimal number from the user
    printf("Enter a decimal number: ");
    scanf("%d", &decimalNumber);

    // Output: Print the equivalent hexadecimal representation
    printf("Hexadecimal equivalent: %X¥n", decimalNumber);

    return 0;
}
```

22. Write a C Program to read a string and count number of vowels in it.

Ans:-

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

int countVowels(char str[]) {
    int count = 0;
    int len = strlen(str);

    for (int i = 0; i < len; ++i) {
        char ch = str[i];

        // Check if the character is a vowel (case-insensitive)
        if (ch == 'a' || ch == 'e' || ch == 'i' || ch == 'o' || ch == 'u' ||
            ch == 'A' || ch == 'E' || ch == 'I' || ch == 'O' || ch == 'U') {
            ++count;
        }
    }

    return count;
}

int main() {
    char inputString[100];

    // Input: Get the string from the user
    printf("Enter a string: ");
    fgets(inputString, sizeof(inputString), stdin);

    // Remove the newline character from the input string
    inputString[strcspn(inputString, "¥n")] = '¥0';

    // Count vowels
    int vowelsCount = countVowels(inputString);

    // Output: Print the count of vowels
    printf("Number of vowels in the string: %d¥n", vowelsCount);

    return 0;
}
```

8. Write a C Program to print transpose of matrix

Ans:-

```
#include <stdio.h>

// Function to print the transpose of a matrix
void printTranspose(int matrix[10][10], int rows, int cols) {
    int transpose[10][10];

    // Computing the transpose
    for (int i = 0; i < rows; ++i) {
        for (int j = 0; j < cols; ++j) {
            transpose[j][i] = matrix[i][j];
        }
    }

    // Printing the transpose
    printf("Transpose of the matrix:¥n");
    for (int i = 0; i < cols; ++i) {
        for (int j = 0; j < rows; ++j) {
            printf("%d¥t", transpose[i][j]);
        }
        printf("¥n");
    }
}

int main() {
    int matrix[10][10];
    int rows, cols;

    // Input: Get the number of rows and columns
    printf("Enter the number of rows: ");
    scanf("%d", &rows);

    printf("Enter the number of columns: ");
    scanf("%d", &cols);
```

```
    // Input: Get the elements of the matrix
    printf("Enter the elements of the matrix:¥n");
    for (int i = 0; i < rows; ++i) {
        for (int j = 0; j < cols; ++j) {
            printf("Enter element at matrix[%d][%d]: ", i, j);
            scanf("%d", &matrix[i][j]);
        }
    }

    // Output: Print the original matrix
    printf("Original matrix:¥n");
    for (int i = 0; i < rows; ++i) {
        for (int j = 0; j < cols; ++j) {
            printf("%d¥t", matrix[i][j]);
        }
        printf("¥n");
    }

    // Print the transpose of the matrix
    printTranspose(matrix, rows, cols);

    return 0;
}
```

52. Write a short note on precedence and order of
evolution.

Ans:- Precedence in programming refers to the order in which operators are evaluated in an expression. It defines the priority or hierarchy of operators. Operators with higher precedence are evaluated before operators with lower precedence. Parentheses can be used to override the default precedence. For example:

```
int result = 2 + 3 * 4; // The multiplication has higher precedence, so it is performed first.
```

47. Write a C language program using recursion n
terms of Fibonacci series.

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

// Function to calculate Fibonacci series term recursively
int fibonacci(int n) {
```

```
    if (n <= 1) {
        return n;
    } else {
        return fibonacci(n - 1) + fibonacci(n - 2);
    }
}

// Function to print the first 'n' terms of the Fibonacci series
void printFibonacci(int n) {
    printf("Fibonacci Series (first %d terms): ", n);
    for (int i = 0; i < n; ++i) {
        printf("%d ", fibonacci(i));
    }
    printf("¥n");
}

int main() {
    int n;

    // Input: Get the number of terms from the user
    printf("Enter the number of terms for Fibonacci series: ");
    scanf("%d", &n);

    // Output: Print the Fibonacci series
    printFibonacci(n);

    return 0;
}
```

46. Write a C language program to define structure for class containing class, name, no. of students and block no. Read 5 records and display it.

Ans:-

```
#include <stdio.h>

// Structure definition for a class
struct Class {
```

```
    char className[50];
    char teacherName[50];
    int numOfStudents;
    int blockNumber;
};

// Function to read information for a class
void readClassInfo(struct Class *class) {
    printf("Enter Class Name: ");
    scanf("%s", class->className);

    printf("Enter Teacher Name: ");
    scanf("%s", class->teacherName);

    printf("Enter Number of Students: ");
    scanf("%d", &class->numOfStudents);

    printf("Enter Block Number: ");
    scanf("%d", &class->blockNumber);
}

// Function to display information for a class
void displayClassInfo(struct Class class) {
    printf("¥nClass Name: %s¥n", class.className);
    printf("Teacher Name: %s¥n", class.teacherName);
    printf("Number of Students: %d¥n", class.numOfStudents);
    printf("Block Number: %d¥n", class.blockNumber);
}

int main() {
    // Declare an array of structures to store 5 records
    struct Class classes[5];

    // Input: Read information for 5 classes
    for (int i = 0; i < 5; ++i) {
        printf("¥nEnter information for Class %d:¥n", i + 1);
        readClassInfo(&classes[i]);
    }
```

```
    // Output: Display information for all 5 classes
    printf("¥nDisplaying information for all classes:¥n");
    for (int i = 0; i < 5; ++i) {
        printf("¥nClass %d:¥n", i + 1);
        displayClassInfo(classes[i]);
    }

    return 0;
}
```

43. Write a C language program to add, list, delete record and modify the current record.

Ans:-

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

// Structure definition for a record
struct Record {
    char name[50];
    int age;
    float salary;
};

// Function to add a new record
void addRecord(struct Record records[], int *count) {
    printf("Enter Name: ");
    scanf("%s", records[*count].name);

    printf("Enter Age: ");
    scanf("%d", &records[*count].age);

    printf("Enter Salary: ");
    scanf("%f", &records[*count].salary);

    (*count)++;
}

// Function to list all records
```

```
void listRecords(struct Record records[], int count) {
    if (count == 0) {
        printf("No records to display.¥n");
        return;
    }

    printf("¥nList of Records:¥n");
    for (int i = 0; i < count; ++i) {
        printf("Record %d:¥n", i + 1);
        printf("Name: %s¥n", records[i].name);
        printf("Age: %d¥n", records[i].age);
        printf("Salary: %.2f¥n", records[i].salary);
        printf("-------------------------¥n");
    }
}

// Function to delete a record
void deleteRecord(struct Record records[], int *count, int index) {
    if (index < 0 || index >= *count) {
        printf("Invalid index for deletion.¥n");
        return;
    }

    // Move records to fill the gap
    for (int i = index; i < *count - 1; ++i) {
        strcpy(records[i].name, records[i + 1].name);
        records[i].age = records[i + 1].age;
        records[i].salary = records[i + 1].salary;
    }

    (*count)--;
    printf("Record deleted successfully.¥n");
}

// Function to modify a record
void modifyRecord(struct Record records[], int count, int index) {
    if (index < 0 || index >= count) {
        printf("Invalid index for modification.¥n");
```

```
        return;
    }

    printf("Enter New Name: ");
    scanf("%s", records[index].name);

    printf("Enter New Age: ");
    scanf("%d", &records[index].age);

    printf("Enter New Salary: ");
    scanf("%f", &records[index].salary);

    printf("Record modified successfully.¥n");
}

int main() {
    struct Record records[50];  // Assuming a maximum of 50 records
    int count = 0;
    int choice, index;

    do {
        printf("¥n1. Add Record¥n");
        printf("2. List Records¥n");
        printf("3. Delete Record¥n");
        printf("4. Modify Record¥n");
        printf("5. Exit¥n");
        printf("Enter your choice: ");
        scanf("%d", &choice);

        switch (choice) {
            case 1:
                addRecord(records, &count);
                break;
            case 2:
                listRecords(records, count);
                break;
            case 3:
                printf("Enter index to delete: ");
```

```
            scanf("%d", &index);
            deleteRecord(records, &count, index - 1);
            break;
        case 4:
            printf("Enter index to modify: ");
            scanf("%d", &index);
            modifyRecord(records, count, index - 1);
            break;
        case 5:
            printf("Exiting the program.¥n");
            break;
        default:
            printf("Invalid choice. Please enter a valid option.¥n");
    }

  } while (choice != 5);

  return 0;
}
```

66. Explain nested structure and self referential structure with example.

Ans:-

```
#include <stdio.h>

// Outer structure
struct Address {
  char city[50];
  char state[50];
};

// Inner structure
struct Employee {
  char name[50];
  int employeeId;
  struct Address empAddress;  // Nested structure as a member
};
```

```
int main() {
    // Declare a variable of the outer structure
    struct Employee emp;

    // Assign values to the members of the outer structure
    strcpy(emp.name, "John Doe");
    emp.employeeId = 101;

    // Assign values to the members of the nested structure
    strcpy(emp.empAddress.city, "New York");
    strcpy(emp.empAddress.state, "NY");

    // Access and print the values
    printf("Employee Name: %s¥n", emp.name);
    printf("Employee ID: %d¥n", emp.employeeId);
    printf("Employee Address: %s, %s¥n", emp.empAddress.city, emp.empAddress.state);

    return 0;
}
```

57. Explain break and continue statements using
syntax and example .

Ans:

break Statement: The break statement is used to terminate the execution of the innermost loop or switch statement in which it appears. When encountered, the control immediately exits the loop or switch, and the program continues with the next statement following the loop or switch.

```
#include <stdio.h>

int main() {
    for (int i = 1; i <= 10; ++i) {
        if (i == 5) {
            printf("Break statement encountered at i = %d¥n", i);
            break;
        }
        printf("%d ", i);
    }

    return 0;
```

```
}
```