PROGRAMMING

1.

```
#include <stdio.h>

struct Student {
    char regNumber[20];
    char firstName[50];
    char lastName[50];
    float programming;
    float networking;
    float accounting;
    float computation;
    float operatingSystem;
};

int main() {
    struct Student student;

    // Input student details
    printf("Enter registration number: ");
    scanf("%s", student.regNumber);

    printf("Enter first name: ");
    scanf("%s", student.firstName);

    printf("Enter last name: ");
    scanf("%s", student.lastName);

    printf("Enter programming marks: ");
    scanf("%f", &student.programming);

    printf("Enter networking marks: ");
    scanf("%f", &student.networking);

    printf("Enter accounting marks: ");
    scanf("%f", &student.accounting);

    printf("Enter computation marks: ");
    scanf("%f", &student.computation);

    printf("Enter operating system marks: ");
    scanf("%f", &student.operatingSystem);

    // Display student details
    printf("\nStudent Details:\n");
    printf("Registration Number: %s\n", student.regNumber);
```

```
    printf("First Name: %s\n", student.firstName);
    printf("Last Name: %s\n", student.lastName);
    printf("Programming Marks: %.2f\n", student.programming);
    printf("Networking Marks: %.2f\n", student.networking);
    printf("Accounting Marks: %.2f\n", student.accounting);
    printf("Computation Marks: %.2f\n", student.computation);
    printf("Operating System Marks: %.2f\n", student.operatingSystem);

    return 0;
}
```

2.
a. Compiler- software tool that translates the source code written in the C programming language into machine code or executable code that can be understood and executed by a computer.

b. Source code - refers to human readable text written by a programmer using c programming language

c.objectcode - refers to the compiled output generated by a compiler from the source code

d. Linkers - refers to a software tool that combines multiple object code files and libraries to create an executable program.

3.
1. Writing the C Program:
First, you need to write the C program in a text editor or an Integrated Development Environment (IDE). In this case, let's consider a simple program to add two numbers:

```c
#include <stdio.h>

int main() {
   int num1, num2, sum;

   printf("Enter two numbers: ");
   scanf("%d %d", &num1, &num2);

   sum = num1 + num2;

   printf("Sum: %d\n", sum);

   return 0;
}
```

2. Preprocessing:

The first step in the compilation process is preprocessing. The preprocessor scans the source code and performs tasks like including header files, macro expansions, and removing comments. In our example, the preprocessor will include the contents of the `stdio.h` header file.

3. Compilation:
After preprocessing, the compiler translates the preprocessed code into assembly language. It checks the syntax, semantics, and type compatibility of the code. If there are any errors, the compiler will report them. If the code is error-free, it generates an object file containing machine code specific to the target platform.

4. Linking:
The next step is linking. If your program uses functions or variables from external libraries or other source files, the linker combines the object file generated in the compilation step with the necessary libraries and resolves references to external symbols. In our example, the linker will link the program with the standard C library.

5. Executable File:
Finally, the linker produces an executable file, which is the machine code that can be directly executed by the operating system. This file contains all the necessary instructions and data to run the program. In our example, the executable file will be created with a default name (e.g., `a.out`).

6. Running the Program:
To run the program, you can execute the generated executable file. In our example, you would run the program, and it would prompt you to enter two numbers. After entering the numbers, it will calculate their sum and display the result.

4.
1. A compiler translates the entire source code into machine code or an intermediate representation before execution.
An interpreter executes the source code line by line without prior translation.

2. A compiler generates an executable file or object code that can be directly executed.
Interpreter does not produce a separate executable file but executes the code directly.

3. Compiled code generally tends to be faster as it is already translated into machine code.
Interpreted code is typically slower as it is translated and executed line by line.

4. Compilers perform a thorough analysis of the source code, catching errors during the compilation process.
Interpreters detect errors as they execute each line of code, stopping at the first encountered error.

5. Compiled programs require more memory as the entire program is loaded into memory before execution.
Interpreted programs use less memory as only the currently executed portion of the code needs to be stored in memory.

6. Compiling a program can take longer, especially for larger projects, as the entire code needs to be compiled before execution. Interpreted languages have a shorter development cycle as changes can be tested immediately without the need for compilation.

5.
1. Arithmetic Operators: Perform basic mathematical operations like addition, subtraction, multiplication, division, and modulus.

2. Relational Operators: Compare values and return a Boolean result (true or false).

3. Logical Operators: Combine Boolean expressions and return a Boolean result. Common logical operators include && (logical AND), || (logical OR), and ! (logical NOT).

4. Bitwise Operators: Perform operations at the bit level. Examples include & (bitwise AND), | (bitwise OR), ^ (bitwise XOR), ~ (bitwise NOT), << (left shift), and >> (right shift).

5. Assignment Operators: Assign values to variables.

6. Increment and Decrement Operators: Increase or decrease the value of a variable by 1.

7. Conditional Operator (Ternary Operator): A shorthand way of writing an if-else statement.

8. Comma Operator:Allows multiple expressions to be evaluated in a single statement. The value of the entire expression is the value of the last sub-expression.