Wairimu Brian Kiarie
Scg200-0240/2018
ICS: Intro to computer programming
Bsc. physical Science

1) a.) A compiler is a program that translates the entire source code written in a high-level language into machine code (object code) all at once. The resulting machine code can be executed independently of the compiler while an interpreter, on the other hand, translates the source code into machine code line by line, executing each line immediately after translation.

   b.) High-level languages are designed to be easily understood by humans and are closer to natural language e.g. Python, Java, and C++. Low-level languages are closer to machine code and are harder for humans to read and write directly e.g. assembly language and machine code.

   c.) Source code is the human-readable version of a program written in a programming language. It contains instructions that can be understood by programmers but not by the computer's processor. Object code is the machine-readable version of the program generated by a compiler or assembler from the source code. It consists of binary instructions that can be executed directly by the computer's processor.

   d.) Data types define the type of data that a variable can hold and the operations that can be performed on that data e.g. integers, floating-point numbers, characters, and boolean values. Variables are used to store data in a program. They are named memory locations whose values can be changed during the execution of the program.

   e.) Printf() is a function used to output formatted data to the console or other output devices. It takes a format string containing placeholders for variables and values to be printed. Scanf() is a function used to read formatted data from the standard input (usually the keyboard) into variables. It takes a format string similar to Printf() to specify the expected format of the input data.

   2. A variable is a named storage location in the computer's memory where data can be stored and manipulated during the execution of a program. Variables have a data type that determines the type of data they can hold, such as integers, floating-point numbers, characters.

   3. i.) Integer Types(int): Represents whole numbers both positive and negative.

   Eg. int age = 25;

   ii.) short: Represents short integers.

   Example: short distance = 1000;

iii.) Floating-Point Types(float): Represents floating-point numbers with single precision. Example: float pi = 3.14;

iv)Character Types(char): Represents single characters.

Example: char grade = 'A';

4. i.) Addition (+): Adds two operands. Example: int sum = 5 + 3;

ii) Subtraction (-): Subtracts the second operand from the first. Example int difference = 10 - 4; // difference will be 6

iii.) Multiplication (*): Multiplies two operands. Example int product = 6 * 2;

iv.) Division (/): Divides the first operand by the second. Example: float quotient = 15.0 / 4.0;

v.) Modulus (%): Returns the remainder of the division of the first operand by the second. Example: int remainder = 15 % 4;

Comparison Operators:

i.) Equal to (==): Checks if two operands are equal. Example:

```
int x = 5;
int y = 5;
if (x == y) {
        // execute this here
}
```

ii.) Not equal to (!=): Checks if two operands are not equal. Example:

```
int a = 10;
int b = 5;
if (a != b) {
        //this exectes
}
```

iii) Greater than (>): Checks if the first operand is greater than the second.

iv.) Less than (<): Checks if the first operand is less than the second.

Combined example:

```
int score = 70;
if (score > 80) {
        // executed if score is greater than 80
} else if (score < 80) {
        // if score is less than 80
} else {
        // executed if score is exactly 80
}
```

v.) >=: Checks if the first operand is less than or equal to the second.

Example:

```
int temperature = 5;
if (temperature <= 10) {
        // This block will be executed
}
```

vi.) Logical AND (&&): Returns true if both operands are true.

```
int num1 = 10;
int num2 = 20;
if (num1 > 0 && num2 > 0) {
    // This block will be executed
}
```

Logical OR (||): Returns true if at least one of the operands is true.

5.

```
#include <stdio.h>
int main() {
    // Define the matrix
    int matrix[3][3];
    // Input values for the matrix
    printf("Enter 9 numbers for the 3x3 matrix:\n");
    for (int i = 0; i < 3; i++) {
    for (int j = 0; j < 3; j++) {
    scanf("%d", &matrix[i][j]);
    }
    }
    // Display the matrix
    printf("The 3x3 matrix is:\n");
    for (int i = 0; i < 3; i++) {
    for (int j = 0; j < 3; j++) {
    printf("%d ", matrix[i][j]);
    }
    printf("\n");
    }
    return 0;
}
```

6.

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

int main() {
    char name[50];
    char course[50];
    char regNumber[20];
    // Input student information
```

7.

```
#include <stdio.h>
int main() {
    int sum = 0;
    // Loop through numbers from 1 to 10
    for (int i = 1; i <= 10; i++) {
```

```
        // Check if the number is even
        if (i % 2 == 0) {
        // If it's even, add it to the sum
        sum += i;
        }
        }
        // Output the sum of even numbers
        printf("The sum of even numbers between 1 and 10 is: %d\n", sum);
        return 0;
}
```