ECE 201 Quiz

Based on Lecture 1, Lecture 2

Valid identifiers follow C rules: they start with a letter or underscore and contain no spaces

Section 1: Conceptual Understanding

Q1. What is software in the context of a computer system?

Software refers to computer instructions or data that can be stored electronically.

Q2. Define machine language. Why is it considered the only language understood by computer hardware?

Machine language is a binary language (0s and 1s) directly understood by computer hardware. It is the only language that hardware can execute without translation.

Q3. What is a source program and how is it transformed into an executable program?

A source program is written in a high-level language. It is transformed into an executable program by compiling into object code and linking with libraries.

Q4. Explain the role of a compiler in program development.

A compiler translates source code into machine code, checks for syntax errors, and prepares the program for execution.

Q5. What is the purpose of the `main()` function in a C program?

The main() function is the entry point of a C program. It contains declarations and executable statements.

Q6. Define a preprocessor directive and give an example.

A preprocessor directive gives instructions to the compiler before actual compilation.

Example: #include <stdio.h> includes standard I/O functions.

Q7. What is the difference between machine language and high-level language?

Machine language is hardware-specific and hard to read; high-level language is human-readable and portable across platforms.

Q8. What does the `#include <stdio.h>` directive do in a C program?

It tells the compiler to include the standard input/output library, enabling functions like printf() and scanf().

Q9. What are comments in C, and how do block comments differ from line comments?

Comments in C are used to document code. Block comments use /* ... */ and can span multiple lines. Line comments use // for single lines.

Q10. Describe the structure of a basic C program using labeled components (e.g., declarations, statements).

A basic C program includes: Preprocessor directives, Global declarations main() function, Local declarations, Statements, Optional additional functions

Q11. What does the computer system consist of?

A computer system consists of hardware, software, data, and users.

Q12. What are the main components of a computer hardware?

Main hardware components include input devices (keyboard, mouse), output devices (monitor, printer), storage devices (hard disk, USB), and processing units (CPU, memory).

Q13. Define computer software and give two examples.

Software refers to computer instructions or data that can be stored electronically. Examples include MS Word, Excel, and library management systems.

Q14. What is the role of a compiler in program execution?

The compiler converts source code into object code, which is then linked to create an executable program.

Q15. Arrange the following in chronological order: Machine Language, Assembly Language, High-Level Language.

Machine Language → Assembly Language → High-Level Language.

Q16. What are the advantages of high-level languages over low-level languages?

High-level languages are easier to learn, portable, and allow focus on logic rather than hardware.

Q17. What is the difference between a source program and an object program?

A source program is written in a high-level language. An object program is its machine-language translation.

Q18. What are the six types of C tokens?

The six types of C tokens are: keywords, identifiers, constants, strings, special symbols, and operators.

Q19. Which of the following are valid identifiers?

- _sum
- 2value
- total_marks
- sum-salary
- INT
- student name

Answer:

Valid identifiers: _sum, total_marks, INT (C is case-sensitive).

Invalid: 2value (starts with digit), sum-salary (contains hyphen), student name (contains space).

Q20. What is the significance of the keyword `int` in C?

The keyword int declares a variable that stores integer values.

Q21. What is the difference between a keyword and an identifier?

A keyword has a fixed meaning in C and cannot be changed. An identifier is a user defined name for variables, functions, etc.

Q22. What does it mean that C is case-sensitive?

Meaning Total and total are treated as different identifiers.

Q23. What is the difference between signed and unsigned integers?

Signed integers can store both positive and negative values.

Unsigned integers store only non-negative values and have a larger positive range.

Q24. What is the precision of `float`, `double`, and `long double`?

float: 6 decimal places, double: 15 decimal places, long double: 19 decimal places

Q25. What is the syntax for declaring multiple variables of the same type?

Syntax: int x, y, z; declares multiple variables of type int.

Q26. What is the difference between a variable and a constant?

A variable is a named memory location whose value can change.

A constant's value cannot change during execution.

Q27. What is the difference between a character constant and a string constant?

A character constant is enclosed in single quotes (e.g., 'A'), while a string constant is enclosed in double quotes (e.g., "Hello").

Q28. Which escape sequence is used for:

- New line
- Tab
- Backslash
- Audible bell
- Carriage return

Answer:

New line: \n , Tab: \t , Backslash: \\ , Audible bell: \a , Carriage return: \r

Q29. What is the purpose of comments in C?

Comments in C are used to document code for human readers. They are ignored by the compiler and help explain logic or clarify sections of code.

Q30. Write a block comment that spans two lines.

```
/* This is a block comment
   that spans two lines. */
```

Q31. What is the syntax of `printf` and `scanf`?

```
printf("format string", data list);
scanf("format string", address list);
```

Q32. What does the format specifier `%d` represent?

%d is used to represent and print an integer value.

Q33. What is the role of the `&` symbol in `scanf`?

The & symbol in scanf is the address operator. It tells the compiler where to store the input value.

Q34. What is the structure of a C program?

A C program typically includes: Preprocessor directives, Global declarations main() function, Local declarations, Executable statements, Optional additional functions

Q35. What is a token in C programming? List the six types of tokens.

A token is the smallest individual unit in a C program.

The six types are:

Keywords, Identifiers, Constants, Strings, Special Symbols, Operators

Q36. Define a keyword and explain why its meaning cannot be changed.

A keyword is a reserved word in C with a fixed meaning that cannot be changed.

Examples include int, float, return.

Q37. What is an identifier in C? List two rules for naming identifiers.

An identifier is a user-defined name for variables, functions, or arrays.

Must start with a letter or underscore.

Cannot contain spaces or hyphens.

Q38. Explain the difference between valid and invalid identifiers with examples.

Valid identifiers follow C rules: they start with a letter or underscore and contain no spaces or symbols.

Invalid ones break these rules—like 2names (starts with digit) or sum-salary (contains hyphen).

Q39. What is a data type in C and why is it important?

A data type defines the kind of data a variable can hold and how much memory it

occupies. It ensures correct interpretation and operations on data.

Q40. Define the `float` data type and describe its precision and size.

float is used for real numbers with 6-digit precision. It occupies 4 bytes.

Q41. What is the difference between signed and unsigned `char` types?

Signed char: range -128 to 127

Unsigned char: range 0 to 255

Q42. What is a constant in C? How does it differ from a variable? Include examples using `const`.

A constant is a fixed value that cannot change during program execution.

Example: const float pi = 3.14;

Unlike variables, constants are immutable.

Q43. Define an escape sequence and give two examples with their meanings.

An escape sequence is a special character combination used for formatting output.

\n = newline

\t = tab

Q44. What does the `scanf()` function do in C? How is it different from `printf()`?

scanf() reads formatted input from the keyboard and stores it in variables.

printf() displays formatted output to the screen.

Q45. Compare machine language, symbolic language, and high-level language in terms of readability, hardware compatibility, and ease of use.

Machine language: binary, unreadable, hardware-specific

Symbolic language: uses mnemonics, slightly more readable

High-level language: human-readable, portable, easier to write

Q46. Why is C considered a structured programming language? What advantages does this offer to developers?

C is structured because it allows modular programming with functions and blocks.

This improves readability, reusability, and maintenance.

Q47. What are the steps involved in creating and running a C program? Briefly describe each phase.

Writing and editing code => Compiling to check syntax and generate object code

=> Linking with libraries => Executing the final program

Q48. How does the compiler differ from the linker in the program development cycle?

Compiler: translates source code to object code

Linker: combines object code with libraries

Q49. What does the error loop represent in the context of compiling and debugging?

The error loop refers to the repeated cycle of compiling, fixing errors,

and recompiling until no errors remain.

Q50. Identify which of the following are valid identifiers and explain why: `student_name`, `2names`, `sum-salary`, `INT MIN`.

Valid: student_name

Invalid: 2names (starts with digit), sum-salary (contains hyphen), INT MIN

(contains space and system-defined value)

Q51. What are the consequences of using a keyword as an identifier in a C program?

Using a keyword as an identifier causes a syntax error because keywords have predefined meanings and cannot be redefined.

Q52. Why is C considered a case-sensitive language? Provide examples to illustrate.

C treats uppercase and lowercase letters as distinct.

Example: Total and total are different identifiers.

Q53. Provide examples of integer, real, character, and string constants. What syntax rules apply to each?

Integer: 123 Character: 'A'

Real: 3.14 String: "Hello"

Q54. Match each data type with its size, range, and precision:

Data Types	Attributes
a. `char`	1. 1 byte, range: -128 to 127
b. `int`	2. 4 bytes, precision: 6 digits
c. `float`	3. 8 bytes, precision: 15 digits
d. `double`	4. 2 or 4 bytes, range: platform-dependent

Your Matches: (Write the letter next to the correct number)

a. char → 1

b. int → 4

c. float → 2 d. double → 3

Q55. Match each format specifier with its corresponding data type:

Format Specifiers	Data Types
a. %d	1. float
b. %f	2. int
c. %c	3. char
d. %lf	4. double

Your Matches: (Write the letter next to the correct number)

a. %d → 2

b. %f → 1

c. %c → 3 d. %lf → 4

Q56. Match each software tool with its role in program development:

Tools

a. Compiler

b. Text Editor

c. Linker

d. Runner

Roles

1. Converts source code to object code
2. Allows writing and editing source code
3. Combines object files into a single executable
4. Executes compiled programs
5. Helps find and fix errors

Your Matches: (Write the letter next to the correct number)

a. Compiler → 1

b. Text Editor → 2

c. Linker → 3 d. Runner → 4

Q57. Match each component with its definition:

Components

a. Compiler

b. Source Program

c. Executable Program

d. Preprocessor Directive

e. main() Function

Definitions

1. The entry point of every C program.
2. A set of instructions written in a high-level language.
3. A translated version of the source code that can be run by the computer.
4. A tool that converts source code into machine code.
5. A command that instructs the compiler to include or modify code before compilation.

Your Matches: (Write the letter next to the correct number)

a. Compiler → 4

b. Source Program → 2

c. Executable Program → 3

d. Preprocessor Directive → 5

e. main() Function → 1

Q58. Match each token type with its description:

Token Types

a. Keyword

b. Identifier

c. Constant

d. Operator

e. Separator

f. String Literal

Descriptions

1. A reserved word with a predefined meaning in C.
2. A name used to identify variables, functions, arrays, etc.
3. A fixed value that does not change during program execution.
4. A symbol that performs operations on variables and values.
5. A character used to separate statements or elements.
6. A sequence of characters enclosed in double quotes.

Your Matches: (Write the letter next to the correct number)

a. Keyword → 1

b. Identifier → 2

c. Constant → 3

d. Operator → 4

e. Separator → 5

f. String Literal → 6

Q59. Match each escape sequence with its meaning:

Escape Sequences	Meanings
a. `\n`	1. Inserts a tab space.
b. `\t`	2. Inserts a newline.
c. `\r`	3. Inserts a carriage return.
d. `\"`	4. Inserts a double quote character.

Your Matches: (Write the letter next to the correct number)

a. \n → 2

b. \t → 1

c. \r → 3 d. \" → 4

Q60. Fill in the blanks with the correct term:

- A machine language is the only language directly understood by computer hardware.
- The directive `#include <stdio.h>` is an example of a preprocessor directive.
- In C, comments that span multiple lines are called block comments.
- A identifier is a meaningful sequence of characters used to name variables, functions, etc.
- The function used to display output in C is printf().
- The sequence of instructions written to perform a specific task is called a program.
- These instructions are formed using special symbols and words according to rigid rules known as syntax.
- Every instruction in C must follow the language's syntax rules.
- Like any other language, C has its own vocabulary and grammar.
- A program written in a high-level language is called a source program.
- The tool used to translate a source program into machine language is called a compiler.
- The translated version of a source program is called an object program.

- The process of combining object code with library code is called linking.
- The only language understood by computer hardware is machine language.
- Every C program must have one function named main().
- The directive `#include <stdio.h>` is an example of a preprocessor directive.
- The keyword `void` means the function does not return any value.
- Statements inside the `main()` function are enclosed within curly braces.
- In C, the smallest individual units are called tokens.
- There are six types of tokens: keywords, identifiers, constants, strings, special symbols, and operators.
- An identifier must begin with a letter or an underscore.
- Identifiers cannot contain spaces or hyphens.
- C is a case-sensitive language.
- The four basic data types in C are char, int, float, and double.
- A variable is a named memory location that stores a value.
- The assignment operator in C is =.
- A character constant is enclosed in single quotes, while a string constant is enclosed in double quotes.
- The keyword `const` is used to declare a constant whose value cannot change.
- The function `printf()` is used for formatted output.
- The function `scanf()` is used for formatted input.
- The symbol `&` is called the address operator and is used to pass the address of a variable.
- The escape sequence `\n` inserts a newline.
- The escape sequence `\t` inserts a tab.

Valid Names		Invalid Name	
a	// Valid but poor style	$sum	// $ is illegal
student_name		2names	// First char digit
_aSystemName		sum-salary	// Contains hyphen
_Bool	// Boolean System id	stdnt Nmbr	// Contains spaces
INT_MIN	// System Defined Value	int	// Keyword

Figure 1: Examples of Valid and Invalid Names

Q61. Using the table of valid/invalid identifiers: Choose two invalid identifiers and explain why they are incorrect.

2names → Invalid because it starts with a digit, which violates the rule that identifiers must begin with a letter or underscore.

sum-salary → Invalid because it contains a hyphen, which violates the rules in C identifiers.

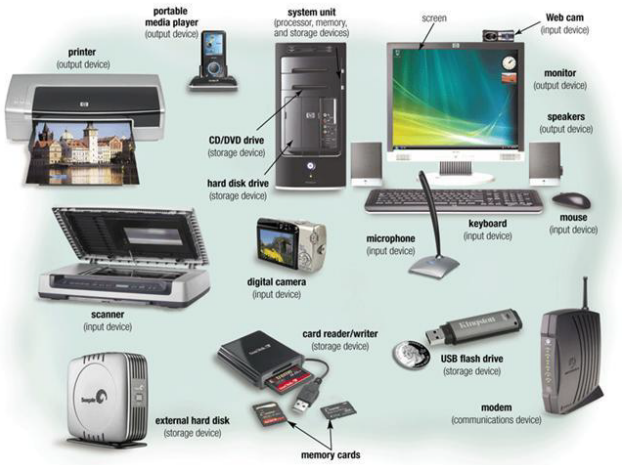

Figure 2: Basic Hardware Components

Q62. Referencing the image of the computer hardware components: Identify and define three input devices and three output devices shown in the diagram.

Input Devices:

Keyboard – Used to enter text, numbers, and commands into the computer.
Webcam – Captures live video or images.
Scanner – Digitizes physical documents or images for electronic use.

Output Devices:

Monitor – Displays visual output from the computer, including text, images, and videos.
Printer – Produces hard copies of digital documents or images on paper.
Speakers – Output audio signals, allowing users to hear sound from the computer.

Section 2: Trace the Code – Predict the Output

T1.
```
int a = 5, b = 2;
int result = a * b + a / b;
printf("%d", result);
```

Output:

12

T2.
```
int x = 10;
x = x + 5;
x = x - 3;
printf("Final value: %d", x);
```

Output:

Final value: 12

T3.
```
char ch = 'A';
printf("%c %d", ch, ch);
```

Output:

A 65

T4.
```
int num = 42;
printf("%5d", num);
```

Output:

42

T5.
```
float pi = 3.14159;
printf("%.2f", pi);
```

Output:

3.14

T6.
```
printf("Hello\tWorld\nGoodbye\rMoon");
```

Output:
Hello World
Moonbye

T7.
```
int a, b;
scanf("%d%d", &a, &b);
// Input: 3 4
printf("%d", a + b);
```

Output:

7

T8.
```
int x = 7, y = 2;
printf("%d", x / y);
```

Output:

3

T9.
```
int a = 1, b = 2, c = 3;
int total = a + b + c;
printf("Sum = %d", total);
```

Output:

Sum = 6

T10.
```
int x;
printf("%d", x);
```

Expected behavior:

unpredictable output or runtime error

Section 3: Debug the Code – Fix the Errors

D1.
```
int a = 5
printf("%d", a);
```

Fix:

```
int a = 5;
printf("%d", a);
```

D2.
```
int x = 10;
y = x + 5;
printf("%d", y);
```

Fix:

```
int x = 10;
int y = x + 5;
printf("%d", y);
```

D3.
```
int age = 25;
printf("Age: %f", age);
```

Fix:

printf("Age: %d", age);

D4.
```
printf("Hello World);
```

Fix:

printf("Hello World");

D5.
```
int num;
scanf("%d", num);
```

Fix:

scanf("%d", &num);

D6.
```
int result = 5 / 2;
printf("%f", result);
```

Fix:

printf("%d", result);

D7.
```
int x;
printf("%d", x);
```

Fix:

x = 0; // or any initialization

Section 4: Extra Questions

C1.
```
#include <stdio.h>

int main() {
    printf("Hello, World!\n")
    return 0;
```

Fix:

```
printf("Hello, World!\n");
return 0;
}
```

C2.
```
#include <stdio.h>

int main() {
    / This prints a message
    printf("Learning C is fun!\n");
    return 0;
}
```

Fix:

```
// This prints a message
```

C3.
```
int age = 25;
float height = 5.9;
char grade = "A";

printf("Age: %f\n", age);
printf("Height: %d\n", height);
printf("Grade: %s\n", grade);
```

Fix:

```
char grade = 'A';
printf("Age: %d\n", age);
printf("Height: %.1f\n", height);
printf("Grade: %c\n", grade);
```

C4.
```
int number;
printf("Enter a number: ");
scanf("%d", number);
printf("You entered: %d\n", number);
```

Fix:

scanf("%d", &number);

C5.
```
printf("Line1/nLine2/tTabbed\nBackslash:-\");
```

Fix:

printf("Line1\\nLine2\\tTabbed\nBackslash: \\");

C6.
```
int a = 5, b = 2;
int result = a * b + a /;
printf("%d", result);
```

Fix:

int result = a * b + a / b;

C7.
```
int x = 150;
char a = 'A';
float pi = 3.14;
printf("%f-%d-%c", x, a, pi);
```

Fix:

printf("%d %c %f", x, a, pi);

End of Quiz — Great work!