Introduction to Basic Fundamentals of Computers

1. **What is the function of an Arithmetic Logic Unit (ALU) in a computer?**
    - **Answer:** The ALU performs arithmetic and logical operations on the data provided by the processor.
2. **Explain the difference between primary and secondary memory.**
    - **Answer:** Primary memory, like RAM, is volatile and directly accessible by the CPU, while secondary memory, like hard drives, is non-volatile and used for long-term storage.
3. **Define what a computer is and mention its basic components.**
    - **Answer:** A computer is an electronic device that processes data according to instructions to produce meaningful output. Its basic components include input devices, output devices, CPU, memory, and storage.
4. **What is the purpose of an operating system?**
    - **Answer:** The operating system manages hardware resources, provides an interface for users, and acts as a platform for running applications.
5. **What are the key functions of the Central Processing Unit (CPU)?**
    - **Answer:** The CPU's key functions are fetching, decoding, executing instructions, and controlling input/output operations.
6. **What is meant by the term "booting" in a computer system?**
    - **Answer:** Booting is the process of starting a computer, during which the system loads the operating system into memory from the hard drive or other storage device.
7. **What is the difference between hardware and software in a computer system?**
    - **Answer:** Hardware refers to the physical components of a computer system, such as the keyboard, monitor, and CPU, while software refers to the programs and instructions that tell the hardware what to do.
8. **Describe the difference between RAM and ROM.**
    - **Answer:** RAM (Random Access Memory) is volatile memory that temporarily stores data and programs currently in use, while ROM (Read-Only Memory) is non-volatile and stores permanent instructions for booting the system.
9. **What is the role of a motherboard in a computer?**
    - **Answer:** The motherboard is the main circuit board of the computer that connects and allows communication between all other components, such as the CPU, memory, storage, and peripherals.

Output type Questions

1. What is the output of the following hexadecimal to binary conversion: 0x1A3?
    - **Output:** 110100011 in binary.
2. Convert the decimal number 345 into its binary form.
    - **Output**: 101011001 in binary.
3. What is the output of the Boolean expression: 1 AND 0 OR 1?
    - **Output:** 1.

4. What will be the output if you try to add the binary numbers 1011 and 1101?
    - **Output:** 11000.
5. Convert the binary number 1111011 to decimal.
    - **Output:** 123 in decimal.
6. What is the ASCII value of the character A?
    - **Output**: 65.

Topics: Introduction to Programming, Identifiers and Constants, Data Types

1. Explain the basic structure of a C program.
2. Why is main() essential, and what are its typical return types?
3. What are preprocessor directives in C? Explain the role of directives like #include and #define.
4. What is a compiler in the context of C programming, and how does it work?
5. What are identifiers in C, and what rules must be followed when naming them?
6. What is the difference between variables and constants in C?
7. Explain how constants are declared and used compared to variables.
8. What are the reserved keywords in C, and why can't they be used as identifiers?
9. What is the scope of an identifier in C?
10. Explain the concept of local and global scope with examples.

11. Discuss the differences between fundamental data types such as int, char, float, and double.

12. What is the difference between high-level and low-level languages?

13. How is a source program converted into an executable?
14. What are the steps to write and execute a C program?
15. What is a compiler and its role in programming?
16. Why are constants preferred over variables in some cases?
17. What's the difference between integer, real, and character constants in C?

Operators

1. Explain the difference between pre-increment (++x) and post-increment (x++) operators in C.
2. What are the shift operators in C, and how are they used?
3. How do the bitwise NOT operator (~) and the logical NOT operator (!) differ in C?

4. What is the purpose of the unary operators in C, and how do they differ from binary operators?
5. Discuss the purpose and behavior of the ternary conditional operator in C.
6. What are the logical operators in C, and how do they differ from bitwise operators?
7. Explain the difference between arithmetic operators and bitwise operators in C.
8. How do the bitwise NOT operator (~) and the logical NOT operator (!) differ in C?
9. Analyze the code snippet

1.

```c
#include <stdio.h>

int main() {
    int a = 10, b = 5;
    int result = a++ + ++b;
    printf("%d", result);
    return 0;
}
```

Output : 16

2.

```c
#include <stdio.h>

int main() {
    int a = 7, b = 3;
    int result = (a % b) * (b + 2);
    printf("%d", result);
    return 0;
}
```

Output : 15

3.

```c
#include <stdio.h>

int main() {
    int x = 8;
    int y = 5;
    int result = x > y ? x : y;
    printf("%d", result);
    return 0;
}
```

Output : 8

4.

```c
#include <stdio.h>

int main() {
    int x = 10, y = 20;
    int result = (x & y) | (x ^ y);
    printf("%d", result);
    return 0;
}
```

Output : 30

5.

```c
#include <stdio.h>

int main() {
    int x = 10;
    int y = x++;
    int z = ++x;
    printf("%d %d %d", x, y, z);
    return 0;
}
```

Output : 12 10 12

6.

```c
#include <stdio.h>

int main() {
    int x = 7; // 0111 in binary
    int y = 14; // 1110 in binary
    int result = x & y;
    printf("%d", result);
    return 0;
}
```

Output : 6

7.

```c
#include <stdio.h>

int main() {
    int a = 12;
    int b = 3;
    int result = (a / b) % 2;
    printf("%d", result);
    return 0;
}
```

Output : 0

8.

```c
#include <stdio.h>

int main() {
    int a = 1, b = 2;
    int result = (a + b) * (a - b) / (a + 1);
    printf("%d", result);
    return 0;
}
```

Output : -3

1.) How does operator precedence affect the following C++ expression?
int result = a + b * c - d / e; how does C decide which operation to perform first? Why is it important to understand this?

2.) Explain the difference between logical AND (&&) and bitwise AND (&) operators in C. Provide an example scenario where each operator would be used.

3.) What is the difference between the equality operator (==) and the assignment operator (=) in C?
4.) What is a bitwise operator in C? List all bitwise operators and explain their purpose with examples.
5.) In C, there are two ways to write an operator that adds 1 to a variable. What are they, and what's the difference between them?

What should be the output of the following code?

```
int main() {

   int a = 11, b = 3;

   int c = a % b;

   printf("c = %d\n", c);

   return 0;

}
```

Expected Output: c = 2

1.) What should be the output of the following code?

```
int main() {
    int x = 5;
    x = (x > 10) ? (x + 1) : ((x < 5) ? (x - 1) : (x * 2));
    printf("%d",x);
    return 0;
}
```

Expected Output: 10

2.) What should be the output of the following code?

```
int main() {
    int a = 7, b = 7, c = 2;
    printf("%d\n", (++a == b) ? c-- : ++b);
    printf("%d\n", (a++ == b) ? c-- : ++b);
    return 0;
}
```

Expected Output: 8
2

3.) What should be the output of the following code?

```
int main() {
    int x = 10, y = 20, z = 5;
    printf("%d", (x < y) || (z > y));
    return 0;
}
```

Expected Output: 1

4.) What should be the output of the following code?

```
int main() {
```

```
    int a = 10, b = 20, c = 30;
    printf("%d", (a < b) ^ (c < b));
    // HINT : Xor operator detects inequality
}
```

Expected Output: 1

5.) What should be the output of the following code?

```
int main() {

        int a = 3, b = 6, c = 4;
        printf("%d", ((a < b) && (b > c)) || ((a > c) || (b < c)));
        return 0;
}
```

Expected Output: 1

6.) What should be the output of the following code?

```
int main() {
    int a = 10, b = 5, c = 2, d = 20, e = 4;
    int result = a + b * c - d / e;
    printf("%d",result);
}
```

Expected Output: 15

7.) What should be the output of the following code?

```
int main() {
    int x = 10, y = 20, z = 5;
    int result = x++ * y-- + ++z / --x;
```

```
    printf("%d",result);
}
```

Expected Output: 200

Decision making & Branching

1. How many case statements can a “switch” statement have?
2. In what scenarios is using a switch statement more appropriate than using if statements?
3. What is the ternary operator in C? Provide an example of its usage.
4. What would happen if there is no default statement in the switch cases?
5. How do you check multiple conditions in a single if statement?
6. How can you immediately terminate a loop or a branching statement.
7. What is the goto statement, and how does it alter the normal flow of control in a program?
8. What will be the output of the following code segment?

```c
int main() {
    int n = 4, a = 1;
    int i, c;
    for (i = 1; i <= n; i++) {
        for (c = 1; c <= i; c++) {
            printf("%d ", a);
            a++;
        }
        printf("\n");
    }
    return 0;
}
```

Output:

1

2 3

4 5 6

7 8 9 10

9. What will be the output of the following code segment?

```c
int main()
{
```

```
    int p = 8, q = 20;
    if (p == 5 && q > 5)
        printf("\nWhy not C");
    else
        printf("\nDefinitely C!");
    return 0;
}
```

Output: Definitely C!

10. What will be the output of the following code segment?

```
int main() {
    int n = 2;
    switch (n) {
        case 1:
            printf("The number is 1\n");
        case 2:
            printf("The number is 2\n");
            break;
        default:
            printf("The number is not 1 or 2\n");
            break;
    }

    return 0;
}
```

Output:

The number is 1

The number is 2

11. What will be the output of the following code segment?

```
int main()
{
    int j = 4, k;
    k = !5 && j;
    printf("\nk = %d", k);
    return 0;
}
```

Output: k=0

12. What will be the output of the following code segment?

```
int main() {
    int m = 1;

    if (m == 1) {
        printf("Delhi");

        if (m == 2) {
            printf("Chennai");
        } else {
            printf("Bangalore");
        }
    } else;

    printf(" END");

    return 0;
}
```

Output: DelhiBangalore END

13. What will be the output of the following code segment?

```
int main() {
    int m;

    for (m = 1; m < 5; m++) {
        printf("%d\n", (m % 2) ? m * 2 : m);
    }

    return 0;
}
```

Output:

2

2

6

4

14. What will be the output of the following code segment?

```
int main() {
    int x = 2;

    if (x == 2 && x != 0);
    {
        printf("\nHi");
        printf("\nHello");
    }
    if(x){
        printf("Bye");
    }

    return 0;
}
```

Output:

Hi

HelloBye

DECISION MAKING AND LOOPS

1. What is the significance of the default case in a switch statement?
2. Differentiate between while and do-while loops.
3. Differentiate between while and for loops.
4. Why might you choose to use a while loop instead of a for loop when iterating over an unknown number of items?
5. Explain the concept of loop control statements (break and continue).
6. Compare the use of switch statements versus if-else if-else chains for handling multiple conditions.
7. What happens if you forget to use break in a switch statement?
8. What is the difference between an entry-controlled loop and an exit-controlled loop? (Compare for/while (entry-controlled) loops with do-while (exit-controlled) loops, focusing on their use cases.)

9. How can you break out of multiple nested loops in C?
   (Discuss strategies like using flags, goto statements, or functions to exit multiple nested loops.)
10. What are the benefits of using compound conditions (e.g., &&, ||) in loop or decision-making statements?

11. Determine the output of the following code:

```
int a = 10;
if (a < 5) {
    printf("Less Than ");
} else if (a == 10) {
    printf("Equal to ");
} else {
    printf("Greater Than ");
}
printf("Decade\n");
```

Expected Output:

```
Equal to Decade
```

12. What will be the output of this code, and how does the `continue` statement affect the loop?

```
for (int i = 0; i < 5; i++) {
    if (i == 3)
        continue;
    printf("%d ", i);
}
```

Expected Output:

```
0 1 2 4
```

13. Analyze the following code and determine the output. Explain the role of the `break` statement here:

```
int i = 0;
while (i < 10) {
    if (i == 5)
```

```
        break;
    printf("%d ", i);
    i++;
}
```

Expected Output:

```
0 1 2 3 4
```

14. What will be the output of this nested loop structure, and how does the control flow work?

```
for (int i = 1; i <= 3; i++) {
    for (int j = 1; j <= 2; j++) {
        printf("%d ", i + j);
    }
}
```

Expected Output:

```
2 3 3 4 4 5
```

15. Determine the output of the following code and explain the effect of logical operators in the `if` condition:

```
int a = 5, b = 10, c = 15;
if (a > b && c > b) {
    printf("Condition 1\n");
} else if (a < b || c < a) {
    printf("Condition 2\n");
} else {
    printf("Condition 3\n");
}
```

Expected Output:

```
Condition 2
```

16. Analyze the following code and predict its output. Explain how the `switch` statement and fall-through behavior work in this context:

```
int x = 3;
switch (x) {
    case 1:
        printf("One ");
    case 2:
```

```
        printf("Two ");
    case 3:
        printf("Three ");
    case 4:
        printf("Four ");
        break;
    case 5:
        printf("Five ");
    default:
        printf("Default ");
}
```

Expected Output:

```
Three Four
```

Explanation:

In this code, the `switch` statement starts executing from the case that matches `x` (which is `3`). Since there's no `break` after case 3, the code "falls through" and continues executing the subsequent cases (`4`), until it encounters a `break` or the end of the `switch` block. Hence, it prints "Three Four".