Concept of Programming (C Language) Answer Schema

Part I: Multiple Choice Questions

1. What is the main purpose of a PAC chart?

   b) To identify input, processing, and output clearly

2. What is debugging?

   d) Fixing errors in a program

3. What type of problem is best solved by computer?

   c) Algorithmic problem

4. Which logic structure executes instructions one after another?

   a) Sequential

5. What is cohesion in modular programming?

   b) A module performs one task only

6. Which type of parameter sends the actual memory address?

   d) Call-by-reference

7. Which loop ensures execution at least once?

   c) do...while

8. What is the result of an infinite loop?

   c) Loop never ends

9. What is the output of this code?

   a) 12

Part II: Answer any FOUR of the following questions.

1. Why is it important to use flowcharts and algorithms before writing code?

Using flowcharts and algorithms before writing code is important because they help in:

- **Planning and designing:** They provide a visual or step-by-step representation of the program's logic, allowing developers to plan the solution before coding.
- **Problem understanding:** They help in clearly understanding the problem and breaking it down into smaller, manageable steps.
- **Error reduction:** By identifying potential logical errors or inefficiencies early in the design phase, they help reduce debugging time later.
- **Communication:** They serve as a clear form of communication among developers, making it easier to collaborate and understand the program's flow.
- **Modifiability and maintenance:** A well-designed algorithm or flowchart makes it easier to modify or maintain the code in the future.

2. How does modular programming improve software development?

Modular programming improves software development by:

- **Reusability:** Modules can be reused in different parts of the same program or in other programs, saving development time and effort.
- **Maintainability:** Changes or updates to one part of the program can be confined to a specific module, reducing the risk of affecting other parts of the code.
- **Readability and understanding:** Breaking down a complex program into smaller, focused modules makes the code easier to read, understand, and debug.
- **Collaboration:** Multiple developers can work on different modules simultaneously, speeding up the development process.
- **Reduced complexity:** It helps manage the complexity of large programs by dividing them into smaller, more manageable units.

3. Why is using loops preferred over repeating code blocks?

Using loops is preferred over repeating code blocks because:

- **Efficiency:** Loops provide a concise way to execute a block of code multiple times without writing the same code repeatedly.

- **Reduced code size:** They significantly reduce the amount of code needed, making programs more compact and readable.
- **Maintainability:** If a change is needed in the repeated action, it only needs to be updated once within the loop, rather than in multiple places.
- **Flexibility:** Loops can be controlled by conditions or iterations, allowing for dynamic execution based on program requirements.
- **Error reduction:** Less repetitive code means fewer opportunities for introducing errors.

4. What is a PAC (Problem Analysis Chart) used for?

A PAC (Problem Analysis Chart) is used to identify input, processing, and output clearly. It helps in organizing the requirements of a system and is often used in the initial stages of system planning.

5. What kind of loop checks its condition after executing the body? How effects if it is false first?

The kind of loop that checks its condition after executing the body is a

do...while loop. If the condition is false the first time it is checked (after the body has executed once), the loop will terminate, and the body of the loop will have executed exactly one time.

6. Differentiate between local and global variables.

- **Local variables:** These are variables declared inside a function or a block of code. Their scope is limited to that specific function or block, meaning they can only be accessed and modified within that region. They are created when the function/block is entered and destroyed when it is exited.
- **Global variables:** These are variables declared outside of any function or block. Their scope is global, meaning they can be accessed and modified from any part of the program after their declaration. They persist throughout the entire execution of the program.

Part III: Write a Program: Choose only THREE questions to answer.

1. Write a program to print the numbers from 1 to 100 using a for loop.

```
#include <stdio.h>
int main() {
   for (int i = 1; i <= 100; i++) {
      printf("%d\n", i);
   }
   return 0;
}
```

2. Write a program that takes an integer input and determines if it is even or odd.

```
#include <stdio.h>
int main() {
   int num;
   printf("Enter an integer: ");
   scanf("%d", &num);
   if (num % 2 == 0) {
      printf("%d is an even number.\n", num);
   } else {
      printf("%d is an odd number.\n", num);
   }
   return 0;
}
```

3. Write a program that accepts three numbers and prints the largest using decision statements.

```
#include <stdio.h>
int main() {
   int num1, num2, num3;
```

```
    printf("Enter three numbers: ");
    scanf("%d %d %d", &num1, &num2, &num3);
    if (num1 >= num2 && num1 >= num3) {
        printf("The largest number is: %d\n", num1);
    } else if (num2 >= num1 && num2 >= num3) {
        printf("The largest number is: %d\n", num2);
    } else {
        printf("The largest number is: %d\n", num3);
    }
    return 0;
}
```

4. Write a program to display all odd numbers between two given numbers using a while loop.

```
#include <stdio.h>
int main() {
    int start, end;
    printf("Enter the starting number: ");
    scanf("%d", &start);
    printf("Enter the ending number: ");
    scanf("%d", &end);
    // Ensure start is odd if it's the first number to be displayed, or increment it
    if (start % 2 == 0) {
        start++;
    }
    printf("Odd numbers between %d and %d are:\n", start, end);
    while (start <= end) {
```

```
        printf("%d\n", start);
        start += 2; // Increment by 2 to get the next odd number
    }
    return 0;
}
```

5. Write a C program that takes two integers as input from the user and displays their sum.

```
#include <stdio.h>
int main() {
    int num1, num2, sum;
    printf("Enter the first integer: ");
    scanf("%d", &num1);
    printf("Enter the second integer: ");
    scanf("%d", &num2);
    sum = num1 + num2;
    printf("The sum of %d and %d is: %d\n", num1, num2, sum);
    return 0;
}
```

Part IV: Fill the blanks.

1. A(n) **algorithm** is a step-by-step procedure used to solve a problem using a computer.
2. The process of finding and correcting errors in a program is known as **debugging**.
3. A(n) **PAC** helps organize input, processing, and output requirements and is often used in system planning.
4. A variable declared inside a module and accessible only within that module is called **local** variable.
5. A(n) **flowchart** is a graphical representation of the logic or process of a program using standard symbols.
6. **UML** diagrams are used in software engineering to visualize the structure and Behavior of a system.