Name: Mani Linkha

Q1. What is the purpose of the main () function in a C program? Explain its significance.

In a C program the main () function is the entry point of program execution. It is where the operating system initiates running code. The importance of main () includes:

1. **Mandatory Function**: All acceptable C programs should have one main () function. It is used without which the program may not be run.
2. **Program Controller**: This sets the initial entry point of program logic, and generally invokes other functions, being the core controller.
3. **Communication Channel:** It provides two-way communication with the operating system:
    - **Input:** It can receive command-line arguments via its parameters (argc and argv).
    - **Output:** It returns an int exit status (e.g., 0 for success, non-zero for error) to report if the program ran successfully or failed.

Q2. Explain the difference between a variable declaration and a variable initialization in C.

In C, the purpose of variable declaration is to inform the compiler the name and the type of a variable. For example, int x; declares a variable x of type integer. At this point, memory is allocated yet the variable contains a garbage value since it has not been assigned any data.

Variable initialization is the assigning of an initial value to a variable during its declaration. For example, int x = 10; both declares the variable and stores the value 10 in it. In this way, declaration merely declares the existence of a non-initialized variable, whereas initialization assigns the variable a certain starting value.

Q4. What are the different data types available in C? Provide examples of each data type.

1. int
    - Description: Used to store whole numbers (both positive and negative).
    - Memory Usage: Typically 4 bytes.
    - Example Values: 1, -10, 0.

      int age = 25;

      int temperature = -5;
2. float
    - Description: Used to store decimal (floating-point) numbers with single precision.
    - Memory Usage: Typically 4 bytes.
    - Example Values: 3.14, -0.5.

      float pi = 3.14;

      float balance = -0.5;
3. double
    - Description: Stores decimal numbers but with higher precision compared to float.
    - Memory Usage: Typically 8 bytes.
    - Example Values: 3.1415926535, -0.123456.

      double precisePi = 3.1415926535;

      double loss = -0.123456;
4. char
    - Description: Stores a single character (letter, digit, or symbol).
    - Memory Usage: 1 byte.

- Example Values: 'A', 'z', '5'.

  char grade = 'A';

  char digit = '5';

5. Bool (or bool when <stdbool.h> is included)
   - Description: Stores logical values — either true or false.
   - Memory Usage: 1 byte.
   - Example Values: true (1), false (0).

     #include <stdbool.h>

     bool isPassed = true;

     bool isEmpty = false;

Q5. Explain the concept of type conversions in C. Provide examples of implicit and explicit type conversions

In C, type conversion simply means changing one kind of value into another so that the program can work smoothly. Sometimes the computer does this for you automatically — this is called implicit conversion. For example, if you add an int (like 10) to a float (like 5.5), the int is automatically turned into a float first, so both match. The computer is just making sure nothing goes wrong and that the result makes sense.

Other times, you may want to force a value to change its type yourself — this is called explicit conversion or type casting. Here, you're basically telling the computer: "Hey, I know what I'm doing, change this now." For example, if you have 9.78 stored in a double and you cast it into an int, it becomes 9 because the decimal part gets dropped. So, implicit conversion is the compiler helping you out, while explicit conversion is you taking control.

Q7. What is the role of the scanf() function in C? Provide an example of its usage.

The scanf() function in C is used to take input from the user. It reads data entered through the keyboard and stores it in the variables provided. The function uses format specifiers (like %d

for integers, %f for floats, %c for characters, %s for strings, etc.) to know the type of data to expect. Since scanf() needs to modify the actual variables, we use the & (address-of operator) to pass the memory location where the input should be stored.

Example:

```
#include <stdio.h>
int main() {
    int age;
    printf("Enter your age: ");
    scanf("%d", &age);
    printf("You entered: %d\n", age);
    return 0;
}
```

Output:

```
Enter your age: 20
You entered: 20
```