1. Why do we need functions? How to define a function? What does a function consist of? What are the naming rules for function names?

Soln:

We need function for some important,

a. modularity: function break down complex task to smaller, manageble parts(module)

b. Code reusability: Once a function is written, it can be used for multiple time for task without rewritting code.

c. Abstraction: It hides the complexity of the implementation, user can focus only what function does.

d. Maintainability : an isolated function easily can read, test and maintain

Defining a function:

Code:

```
return_type  function_name ( list of variable with data type )
{
        // function body
}
```

A function consist of following things:

a. Return type:

b. Function name

c. Parameter (optional)

d. Function body

e. Return statement (optional)

Naming rule for functions:

a. Start with letter or underscore(_)

b. Contains only letter,digits and underscore

c. Can not be C keyword

d. Follow naming convention

2. Why do we need function declaration? If there is no function declaration, sometimes the compiler will report an error, sometimes report a warning, sometimes not any error or warning. Why is this? Illustration separately

Soln: Function declaration is necessary to inform compiler about the function before it actual defination appear in the code.

Without function declaration, the behavior of the program depends where and how the function is used before its definition.

a. **Function defined before use**: No issue.

Code:

```
#include<stdio.h>

int add(int a, int b){

	return a + b;

}

int main(){

	int result = add(5, 3);

	printf(“Sum = %d\n”, result);

}
```

b. **Function is used before definition:** C89 or older version compile with warning but still work, because compiler assumes return type int.

But modern C compiler show error, because implicit function declaration not allowed in modern compiler.

Code:

```
#include<stdio.h>

int main(){

	int result = add(5, 3);

	printf(“Sum = %d\n”, result);
```

```
}

int add(int a, int b){

	return a + b;

}
```

c.Mis match return type or parameter:

If function is called before declaration and return type or parameter does not match, then it shows error modern and older compiler.

Code:

```
#include<stdio.h>

int main(){

	int result = add(5, 3);

	printf(“Sum = %d\n”, result);

}

float add(int a, int b){

	return a + b;

}
```

3. What are the matters that need attention for function return values?

When working with function in c, handlig return value is crucial for avoid bug.Here are some points to consider:

a. Return type must match the function defination otherwise it shows undefined behavier
b. Mismatch return type may cause implicit Conversion
c. Returning local variable leads to undefined behavier

d. Returning Void for No Return Value
e. Function that no return value.
f.

4. What is an inline function? What is the form of it?

An inline function is a function that the compiler expands at the point of call instead of performing a regular function call.

Form:

```
inline return_type function_name(parameters){

// function body

}
```

5. What is the difference between inline function, normal function, and macro function? What are their advantages and disadvantages?

Inline function use inline keyword with normal function. Normal function is A standard function with a separate memory location., macro function is a preprocessor directive using #define with no type checking.

Advantage for inline function:

a. Removes function call overhead, Faster execution for small functions.

b. Type safety is maintained.

c. Works like a normal function but avoids extra memory usage.

Disadvantage :
a. Not guaranteed: The compiler may ignore the inline request.
b. Code bloat if used with large functions

Normal function:
Advantage:
a.Suitable for large functions and reduces code duplication.
b.Easy to debug (appears in call stack).
c.Supports recursion.

Disadvantage:
a.Has function call overhead (pushing/popping registers in the stack).
b.Slightly slower than inline and macros

Macro Function:
Advantages:
a. Fastest execution (code is expanded before compilation).
b. No function call overhead.
c. Can be used in preprocessor directives.

Disadvantages:
a. No type checking
b. Hard to debug
c. Code bloat if used with large functions