Summary: Structs, Unions, Enums, and Typedef in C

Problem 1 – Struct with Bit Fields

- A struct student_info was created to store compact student data using bit fields.
- Bit fields allow specifying exact bit sizes for fields (e.g., unsigned int age : 8).
- Bit fields are useful for memory optimization, but have limitations:
  - The base type size restricts them.
  - Overflow occurs if a value exceeds the allocated bit size.
- The total size of a struct with bit fields includes padding due to alignment.

Problem 2 – Using typedef for Primitive Types

- typedef was used to create aliases for primitive data types (e.g., typedef int i_int;).
- The goal is improved readability and clarity, especially in larger programs.
- The sizes of these typedefs remain the same as their underlying primitive types.
- A naming scheme was used to indicate the original type (e.g., i_, f_, c_).

Problem 3 – Complex Numbers with typedef struct

- A typedef struct was used to define a complex number type with real and imaginary parts.
- A function was written to add two complex numbers by passing structures.
- This demonstrates how typedef simplifies code reuse and improves clarity when working with custom types.

Problem 4 – Enum for Days of the Week

- An enum Day was created to represent the days of the week from SATURDAY to FRIDAY.
- A function was implemented to determine whether a given day is a weekday or weekend.
- The default underlying type of an enum in C is typically int (4 bytes).

Problem 5 – Enum with ASCII Operation Codes

- An enum Operation was defined using ASCII values to represent operations such as +, -, *, /.
- A calculator was implemented that takes two operands and an operator, using a switch statement with enum values.
- The size of an enum is generally 4 bytes. Adding more members does not change its size unless values require a larger base type.

Problem 6 – Using a Union for Multiple Data Types

- A union was created with members: int, float, and char.

- Demonstrated how only one member can be used at a time, since all share the same memory.
- Assigning a value to one member affects the others due to memory overlap.
- Unions are more memory-efficient when only one of several values is needed at any time.

Key Concepts and Guide Answers

What is the purpose of typedef?

To create type aliases that improve code readability and simplify complex type definitions.

How are bit fields declared, and what are their limitations?

Declared using a colon and number of bits (e.g., int x : 4;). They are limited by the size of the base type and may not be portable across compilers.

What happens if a bit field overflows?

The value is truncated or wraps around, potentially leading to incorrect data storage.

How is typedef used with complex types like structs and unions?

It is used to rename complex structures for easier reuse. For example:

c

CopyEdit

```
typedef struct {
   int x;
   int y;
} Point;
```

What is the default underlying type of an enum?

By default, an enum uses the int type, which is typically 4 bytes.

How is a union different from a struct?

In a struct, each member has its own memory.
In a union, all members share the same memory, and only one can hold a value at a time.

When is using a union more memory-efficient?

When you only need to store one of several possible data types at a time. It minimizes memory usage by overlapping storage.