Chapter 1 Answers

1.
```
struct Student {
   int rollNo;
   char name[50];
   float marks;
};
```

   Advantages:
   - Combines related data types under one name.
   - Easier data management and code readability.
   -

2. Nested Structure
3. By properly deallocating dynamic memory using the delete operator.
4. Problems:
   - The program uses more and more memory each time it runs.
   - This can slow down the computer or even crash the program.
5. (a) Using delete operator.
   (b)
```
struct Time {
    int hour;
    int minute;
    int second;
  };
  Time current_time, next_time;
```

6.
```
#include <iostream>
using namespace std;
struct Student {
   char name[50];
   int roll;
   float marks;
};
int main() {
   Student s;
   cout << "Enter name, roll, marks: ";
   cin >> s.name >> s.roll >> s.marks;
   cout << "Student Info:\n";
   cout << "Name: " << s.name << "\nRoll: " << s.roll << "\nMarks: " << s.marks;
   return 0;
}
```

7.

```
#include <iostream>
using namespace std;
int main() {
    int n;
    cout << "Enter number of students: ";
    cin >> n;
    int *marks = new int[n];
    cout << "Enter marks:\n";
    for (int i = 0; i < n; i++) cin >> *(marks + i);
    for (int i = 0; i < n-1; i++)
        for (int j = 0; j < n-i-1; j++)
            if (*(marks + j) < *(marks + j + 1))
                swap(*(marks + j), *(marks + j + 1));
    cout << "Marks in descending order:\n";
    for (int i = 0; i < n; i++) cout << *(marks + i) << " ";
    delete[] marks;
    return 0;
}
```

8.

Array	**Structures**
Same type only	Different types allowed
Contiguous	May not be contiguous
Indexed	Accessed using member names

9. new
10.

```
char* months[12] = {
    "January", "February", "March", "April", "May", "June",
    "July", "August", "September", "October", "November", "December"
};
```

11. This is called a nested structure. It helps in grouping logically related data.
12. (iv)*ptr will give the value of the variable ‘num’.

13.

Static Allocation	**Dynamic Allocation**
It takes place before the execution of the program	It takes place at the time of execution of the program
Ordinary variable is used	Pointer is essential

14.
```
Struct //Structure tag is missing
{
   int regno;
   char name[20];
   float mark = 100;
};
```

15.

Static Allocation	**Dynamic Allocation**
It takes place before the execution of the program	It takes place at the time of execution of the program
Operating system takes the responsibility of allocation variable declaration statement	New operator is required
Ordinary variable is used	Pointer is essential
Data is referenced using variable	Data accessed using direction operator only
No statement is needed for deallocation	Delete operator is used for deallocation

16.
```
cout << *p;       // 5
cout << *p + 1;   // 6
cout << *(p + 1); // 10
```

17. Self-referential structure is a structure in which one of the element is a pointer to the same structure. A location of this type contains data and address of another location of same type.

18.
```
int *ptr = new int(10);   // Allocates a single integer variable with value 10.
int *ptr = new int[10];   // Allocates an array of integers with size 10.
```

19. Pointer is a variable that can hold the address of another memory location.
```
char* country = "India";
```

20.
```
struct Time {
    int hour;
    int minute;
    int second;
};
```

21.
```
cout << *(p + 2);    // 20
cout << *p + 3;     // 13
```

22.

Static Allocation	Dynamic Allocation
It takes place before the execution of the program	It takes place at the time of execution of the program
Operating system takes the responsibility of allocation variable declaration statement	New operator is required
Ordinary variable is used	Pointer is essential
Data is referenced using variable	Data accessed using direction operator only
No statement is needed for deallocation	Delete operator is used for deallocation

23. (a) complex c1;
    (b) c1.real = 15;

24. (*) It is used Only with the pointers and it retrieves the value pointed by the pointer.
    (&) It is used to get the address of the variable.

25.

Array	Structure
It is a derived data type	It is a user defined Data type
Collection of same type of data	Collection of different types of data
Elements of the array are accessed using an index	Elements of the structure is accessed using dot (.) operator
Array of structures is possible	Structure can contain array as a element

26. New

27. Pointer is a variable that can hold the address of another memory location.
    Syntax: Datatype* variable;
    Eg: int *ptr;

28. Static -> Compile time and use fixed memory.
    Dynamic -> Run time and not fixed memory.

29. Structure is a user defined data type. it is the collection of logically related data items which may be different types under a common name.
    Structure -> Can store different data items and elements accessed by names.
    Array -> Store only same type and elements accessed by index.