(e) Consider the following code snippet:

```
struct emp
{
    int a ;
    char ch ;
    float s ;
} ;
struct emp e ;
printf ( "%u %u %u", &e.a, &e.ch, &e.s ) ;
```

If we execute this program using TC/TC++ compiler we get the addresses as:

```
65518 65520 65521
```

As expected, in memory the **char** begins immediately after the **int** and **float** begins immediately after the **char**.

However, if we run the same program using VC++ compiler then the output turns out to be:

```
1245044 1245048 1245052
```

It can be observed from this output that the **float** doesn't get stored immediately after the **char**. In fact there is a hole of three bytes after the **char**. Let us understand the reason for this. VC++ is a 32-bit compiler targeted to generate code for a 32-bit microprocessor. The architecture of this microprocessor is such that it is able to fetch the data that is present at an address, which is a multiple of four much faster than the data present at any other address. Hence the VC++ compiler aligns every element of a structure at an address that is multiple of four. That's the reason why there were three holes created between the **char** and the **float**.

However, some programs need to exercise precise control over the memory areas where data is placed. For example, suppose we wish to read the contents of the boot sector (first sector on the floppy/hard disk) into a structure. For this the byte arrangement of the structure elements must match the arrangement of various fields in the boot sector of the disk. The **#pragma pack** directive offers a way to fulfill this requirement. This directive specifies packing alignment for structure members. The pragma takes effect at the first structure declaration after the pragma is seen. Turbo C/C++ compiler doesn't support this feature, VC++ compiler does. The following code shows how to use this directive.

```
#pragma pack(1)
struct emp
{
    int a ;
    char ch ;
    float s ;
} ;
```

```
#pragma pack( )

struct emp e ;
printf ( "%u %u %u", &e.a, &e.ch, &e.s ) ;
```

Here, **#pragma pack (1)** lets each structure element to begin on a 1-byte boundary as justified by the output of the program given below:

```
1245044  1245048  1245049
```