Ans 1 Row-major and column-major order are methods for storing multidimensional arrays in linear storage as random access memory

eg: $\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$ Row-major : 1 2 3 4
Col-major : 1 3 2 4

⇒ In Column-major

LOC (A[i, j]) = base address + W * [M*j + i]

W = word size = no. of bytes occupied by each element

M = no. of Rows

⇒ In Row-Major

LOR (A[i, j]) = base address + W * [N*i + j]

N - no. of columns

Ans 2 Row major is used in C/C++, Pascal, ..
Col major is used in Fortran, Matlab, etc.

- An alternative way to store multidimensional arrays are storing pointers to rows in contiguous location, but not the rows themselves.

So they are neither row major or column-major.

Ans 3

- There is no possibility of shift/reduce conflict
- There is RR conflic as the parser can't decide whether tor reduce a by C or a by S.

$\ast$ First (A) = $\{a, c, \epsilon\}$ Follow (A) = $\{+\}$

First (B) = $\{a, c, \epsilon\}$ Follow (B) = $\{+\}$

First (C) = $\{a\}$ Follow (C) = $\{+\}$

First (D) = $\{c, \epsilon\}$ Follow (D) = $\{e\}$

First (S) = $\{a, e, \epsilon\}$ Follow (S) = $\{+\}$

- There is first first conflict as, $S \rightarrow \underset{\smile}{A}$ and $S \rightarrow a$ both have First (A) = a
- There is first follow conflict as First(D) and Follow (D) both contain 'e'.

Ans 4

- A descriptor is the collection of the attributes of a variable.
- Descriptors are used for type-checking and building the code for allocation and deallocation operations
- A compile-time descriptor for multidimensional array.

Multidimensional array
Element Type
Index Typepe
Numer of dimension
Index Range 1
⋮
Index Range n
address

Ans 5

using if :- bool b=0

```
    if (i==1 || b==1) { # code for case 1; b = 1; }
else if (i==2 || b==1) {# -- for case 2; b=1; }
        .
        .
        .
```

using Goto :-

```
        ~~bool b=0~~;
    if (i==1) goto case 1;
    if (i==2) goto case2;
        .
        .

    Case1 :    # code


    Case 2:   # code


    Case 3:  # code
```

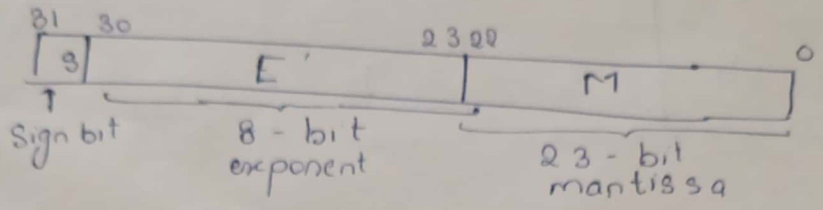
Ans ·6 Yes, the union will save space but the value of data.i and data.f will be corrupted as union can only hold one value at a time :

Correction :-

```
union Data data;
  data . i = 10;
  printf ( "%d" , data.i);
  data.f = 220. 5;
   print (data.f);
  data .str = " C Programming";
  print (dat. str):
```

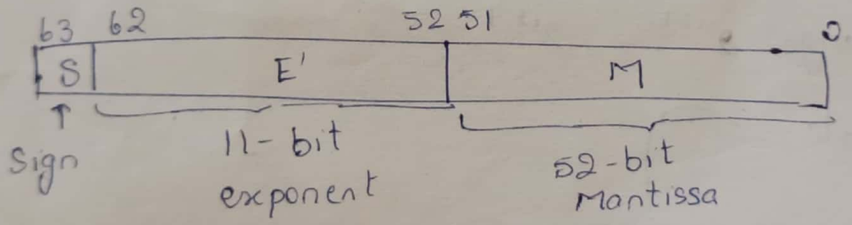
Ans 7

1] Single :-



- Bias = ~~127~~ 127 (excess - 127)
- Range : $2^{-126} - 2^{127}$

2] Double :-



- Bias = 1023 (excess - 1023 format)
- Range :- $2^{-1022} - 2^{1023}$

eg :- ~~1259.1~~

Eg $(1259.125)_{10}$

1) Single :-

$1259 = 100\ 1110\ 1011$

$0.125 = 0.001$

$1259.125 = 100\ 111\ 0\ 1\ 0\ 11 . 001$

$= 1.00\ 111\ 01\ 011001 \times 2^{10}$

$S = 0, \quad E = 10 \quad , M = (00\ 111\ 010\ 11\ 001)_2$

Bias = 127

$E' = E + 127 = (137)_{10} = (10001001)_2$

$$\Rightarrow \underbrace{0}_{\text{sign}} \ \underbrace{10001001}_{\text{exp.}} \ \underbrace{0011101011001\ 00000 --- 0}_{\substack{\text{mant.} \\ (23\text{-bit})}}$$

2) Double :-

$S = 0, \ E = 10 \ , \ M = (00\ 111\ 0\ 10\ 11\ 0\ 0\ 1)_2$

Bias = 1023

$E' = 10 + 1023 = 1033_{10} = (1\ 000000\ 1001)_2$

$$\Rightarrow \underbrace{0}_{\text{sign}} \ \underbrace{10000001001}_{\text{Exponent}} \ \underbrace{0011101011001\ 000 - - 0}_{\substack{\text{mantissa} \\ (52\ \text{bit})}}$$