STORAGE CLASSES IN C (2014)

Variables used in a C programming are stored in RAM or CPU memory register. There are four storage classes into which the variables are declared and stored.

1. **auto** storage class
2. **static** storage class
3. **extern** storage class
4. **register** storage class

Depending on the type of the storage class, the scope and behaviour of the variables in a C program.

auto or Automatic Storage : [2016]

Variables declared in this class are stored in RAM. This is the default storage class and the keyword **auto** is used to declare variables. Auto variables are active in a block in which they are declared. Note that a block means the statements inside the braces { }.

[**auto** storage class is commonly used in all C programs without the keyword **auto.**]

static Storage Class : [2016]

Variables declared in this class are also stored in the RAM. The keyword **static** is used to declare these variables. Similar to auto variables, the static variables are also active in the block in which they are declared, and they retain the latest value. The static variables are commonly used along with functions.

extern Storage Class: [2016]

Global variables are declared using this class and they are stored in the RAM. The keyword **extern** is used to declare these variables. Note that the global variables are also declared outside the **main()** function. **extern** class can be used to consider a local variable in a block as a global variable.

register Storage Class:

Variables declared using this class are stored in the CPU memory register. The keyword **register** is used to declare these variables. Only a few variables which are frequently used in the program are declared using this class to improve the program execution speed.

The behaviour of register variables is similar to that of **auto** variables except that their storage locations are

different. In case of non-availability of CPU memory register, these variables are stored in RAM as **auto** variables.

Example : Write a function to multiply A matrix of order m×n with B matrix of order n×1 and write the main program to input values and output the resultant matrix.

Ans : /* matrix multiplication using functions*/

```
#include <stdio.h>
int a[10] [10], b[10] [10], c[10] [10], m,n,l,i,j,k;
main()
{
  extern int a[10] [10],b[10] [10], c[10] [10], m,n,l;
  void matmul();
  printf("\n Enter order of A matrix (m×n) :");
  scanf("%d %d", &m, &n);
  /*loop to read all values*/
  printf("\n Enter A matrix values \n");
  for(i=0; i<m; i++)
    for(j=0; j<n; j++)
```

```
        scanf("%d", &a[i] [j]);
    printf("\n Enter order of B matrix (n×1) :");
    scanf("%d%d", &n, &1);
    printf("\n Enter B matrix values \n");
      for(i=0; i<n; i++)
        for(j=0; j<1; j++)
           scanf("%d", &b[i] [j]);
    /*call function*/
    matmul();
    printf("\n Resultant matrix is \n");
    for(i=0; i<m; i++)
       {
          for(j=0; j<1; j++)
              printf("%5d", c[i] [j]);
           printf("\n");
          }
     getch ();
     }
/*function to multiply matrices*/
```

```
void matmul()
{
  extern int a[10] [10], b[10] [10], c[10] [10], m,n,l;
  for(i=0; i<m; i++)
    for(j=0; j<1; j++)
     {
        c[i] [j] =0;
        for(k=0; k<n; k++)
          c[i] [j] = c[i] [j] + a[i] [k] * b[k] [j];
     }
 }
```