What do you mean by Static Variable?

There are three main uses for the static:

1. If it declare within a function: it retains the value of the variable between different function calls.
2. If it is declared for a function name: by default functions are extern…so it will be visible from other files, but if the function declaration is static…then it is invisible for outer files.
3. Static for global variables: by default we can use the global variables from out side files, but if it is static global variable then it is limited to the file where it is declared.

What is Command Line Arguments

```
void main(int argc , char * argv[])
        {
           int x;
           printf ("total no of arguments passed:%d",argc-1);
           for(x=0;x<argc;x++)
           printf("\n%s", argv[x]);
           getch();
         }
```

Find the addition of numbers using Command Line Arguments

```
void main(int argc , char * argv[])
        {
          int x , y=0;
          for(x=0;x<argc;x++)
          y = y + atoi (argv [x]);
          printf("sum = %d",y);
          getch();
        }
```

Enumerated Data Type:

- The enumerated data type gives us an opportunity to invent our own data type and define what values the variable of this data type can take.

```
enum Boolean
    {
        true, false
    };
enum Boolean flag1,flag2;
```

1. Here the first part declares the data type and specifies its possible values. These values are called 'enumerators'.
2. The second part declares variables of this data type.

How would you dynamically allocate a 1-D array of integers

```
#include"alloc.h"
#define MAX 10
void main()
   {
       int *x,y;
       x=(int *) malloc (MAX * sizeof (int));
       for(y=0; y< MAX; y++)
          {
             x[y]=y;
             printf("\n%d", x[y]);
          } }
```

How would you increase the size of a dynamically allocated 1-D array of integers

```
#include"alloc.h"
void main()
   {
      int *p, *t;
      p=(int *) malloc  (20);
      t = p;
      t =(int *) realloc  (p,40);
      if (t == NULL)
         printf ("can't reallocate");
      else
         {
           free (p);
            p = t;
         }
   }
```