What is a Macro?

- A macro is a preprocessor directive that provides a mechanism for token replacement in our source code. Macros are created by using the #define statement.
- **What is the output of the following program?**

```
#define cube(x) (x*x*x)
void main()
  {
     printf("%d",cube(1+2));
   }
```

Output: 7

What is a Macro?

- A macro is a preprocessor directive that provides a mechanism for token replacement in our source code. Macros are created by using the #define statement.
- **What is the output of the following program?**

```
#define cube(x) (x*x*x)
void main()
  {
     printf("%d",cube(1+2));
   }
```

Output: 7

What is function prototype?

- Function prototype tells the compiler:
  - the name of the function
  - the type of the output returned by the function
  - the total no of arguments received by the function

Ex: int add(int, int);

Return type Name of the function Parameter list

There are mainly four types of functions:

1. Function with no arguments and no return values
2. Function with arguments and no return values
3. Function with arguments and one return values
4. Function with no arguments but return a value
5. Function that returns multiple values

Function with no arguments and no return values

```
#include<stdio.h>
 void printsum(); // function
                          declaration
void main()
   {
     printsum();
   }
 void printsum() // function
                          definition
   {
     printf("Sum of 2 and 3
                              is:%d", (2+3));
   }
```

Function with arguments and no return values

```
#include<stdio.h>
 void add(int, int); // function
                          declaration
void main()
   {
     add(2,3);
   }
 void add(int a, int b) // function
                                definition
   {
     int c;
     c=a+b;
     printf("Sum of 2 and 3
                                  is:%d", c);
   }
```

Function with arguments and one return values

```
#include<stdio.h>
 int add(int, int); // function
                          declaration
void main()
    {
     int r;
     r = add(2,3);
      printf("Sum of 2 and 3
                                is:%d", r);
    }
 int add(int a, int b) // function
                                definition
    {
       return (a+b);
    }
```

Function with no arguments but return a value

```
#include<stdio.h>
 int get_number(void); // function
                              declaration
void main()
    {
     int r;
     r = get-number();
      printf("Number= %d", r);
    }
 int get_number(void) // function
                                    definition
    {
      int no;
      scanf("%d",&no);
      return (no);
    }
```

Write your own String length function using pointer

```
main( )
  {
  char  arr[ ] = “Kolkata" ;
  int  len ;
  len = my_strlen ( arr ) ;
  printf ( "\nstring = %s ,
 length = %d", arr, len) ;
  }
```

```
my_strlen ( char  *s )
   {
      int  length = 0 ;
      while ( *s != '\0' )
       {
          length++ ;
          s++ ;
       }
      return ( length ) ;
  }
```

Write your own String copy function using pointer

```
main( )
 {
  char  source[ ] = “Hello" ;
  char  target[10] ;
 my_strcpy ( target, source ) ;
  printf ( "\n source string =
 %s", source ) ;
  printf ( "\n target string =
 %s", target ) ;
 }
```

```
my_strcpy ( char  *t, char  *s )
        {
             while ( *s != '\0' )
                {
                *t = *s ;
                 s++ ;
                 t++;
                }
         *t = '\0' ;
        }
```

Write your own String concatenates function using pointer

```
main( )
  {
   char  source[ ] = “Hello" ;
   char  target[ ] =“Hi”;
    my_strcat( target, source ) ;
   printf ( "\n target string = %s", target ) ;
  }
```

```
my_strcat ( char  *t, char  *s )
        {
          while ( *t != '\o' )
          t++;
            while ( *s != '\o' )
              {
              *t = *s ;
               s++ ;
               t++;
              }
         *t = '\o' ;
        }
```