1. memcpy

Copies **n** characters from memory area **src** to memory area **dest**.

```
void *memcpy(void *dest, const void * src, size_t n)
```

```
  const char src[50] = "http://www.tutorialspoint.com";
   char dest[50];
   strcpy(dest,"Heloooo!!");
   printf("Before memcpy dest = %s\n", dest);
   memcpy(dest, src, strlen(src)+1);
```

2. strcpy

Copies the string pointed to, by **src** to **dest**.

```
char *strcpy(char *dest, const char *src)
```

```
  char src[40];
  char dest[100];

   memset(dest, '\0', sizeof(dest));
   strcpy(src, "This is tutorialspoint.com");
   strcpy(dest, src);
```

3. memset

Copies the character **c** (an unsigned char) to the first **n** characters of the string pointed to, by the argument **str**.

```
void *memset(void *str, int c, size_t n)
```

```
  char str[50];
  ...
   memset(str,'\0',sizeof(str));
```

4. scanf

Reads formatted input from stdin.

Reads up to the 1 st space, tab or newline (non included). The rest remains in the input buffer.

```
int scanf(const char *format, ...)
```

```
  char c;
  char arr;
  ...
   scanf(a,'\0',sizeof(str));
    scanf("%c", &c);
    scanf("%s", arr);
```

5. sprintf

Sends formatted output to a string pointed to, by **str**.

```
int sprintf(char *str, const char *format, ...)
```

```
   char str[80];
   sprintf(str, "Value of Pi = %f", M_PI);
```

6. fgets

Reads a line from the specified stream and stores it into the string pointed to by **str**. It stops when either **(n-1)** characters are read, the **newline** character is read, or the **end-of-file** is reached, whichever comes first.

If a newline is read, it is stored into the buffer.

A terminating null byte ('\0') is stored after the last character in the buffer.

If the End-of-File is encountered and no characters have been read or a error occurs, the contents of str remain unchanged and a **null** pointer is returned.

```
char *fgets(char *str, int n, FILE *stream)
```

```
int main () {
   FILE *fp;
   char str[60];

   /* opening file for reading */
   fp = fopen("file.txt" , "r");
   if(fp == NULL) {
      perror("Error opening file");
      return(-1);
   }
   if( fgets (str, 60, fp)!=NULL ) {
      /* writing content to stdout */
      puts(str);
   }
   fclose(fp);

   return(0);
}
```

7. puts

The C library function writes a string to stdout up to but **not including the null character**. A **newline** character is **appended** to the output.

```
int puts(const char *str)
```

8. fputs

Writes a string to the specified stream up to but **not including the null character.**

```
int fputs(const char *str, FILE *stream)
```

```
    FILE *fp;
   fp = fopen("file.txt", "w+");

   fputs("This is c programming.", fp);
   fputs("This is a system programming language.", fp);

   fclose(fp);
```

9. getchar

Gets a character (an unsigned char) from stdin.

```
int getchar(void)
```

```
   char c;
   printf("Enter character: ");
   c = getchar();
```

10. getline

Reads an entire line from stream, storing the address of the buffer containing the text into ***lineptr.**

Buffer is **null-terminated** and includes the **newline** character, if one was found.

***lineptr** can contain a pointer to a malloc(3)-allocated buffer ***n bytes** in size.

```
int main(int argc, char *argv[]){
   FILE *stream;
   char *line = NULL;
   size_t len = 0;
   ssize_t nread;

   if (argc != 2) {
     fprintf(stderr, "Usage: %s <file>\n", argv[0]);
      exit(EXIT_FAILURE);
   }

    stream = fopen(argv[1], "r");
     if (stream == NULL) {
        perror("fopen");
        exit(EXIT_FAILURE);
     }

     while ((nread = getline(&line, &len, stream)) != -1) {
       printf("Retrieved line of length %zu:\n", nread);
       fwrite(line, nread, 1, stdout);
     }

     free(line);
     close(stream);
     exit(EXIT_SUCCESS);
 }
```