ASSIGNMENT-10

Q1.Write a user define functions for ::

a. mystrcpy

b. mystrlen

c. mystrcmp

d. mystrcat

e. mystrncpy

f. mystrupper

g. mystrlower

h. mystrrev

i. mystrstr

j. mystrcasecmp

k. mystrchr

l. mystrrchr

m. mystrncmp

n. mystrnstr

o. mystrncat

p. mystrncasecmp

```
#include <stdio.h>
#include <string.h>
// Function prototypes
char* mystrcpy(char* dest, const char* src);
size_t mystrlen(const char* str);
int mystrcmp(const char* str1, const char* str2);
char* mystrcat(char* dest, const char* src);
char* mystrncpy(char* dest, const char* src, size_t n);
char* mystrupper(char* str);
char* mystrlower(char* str);
char* mystrrev(char* str);
char* mystrstr(const char* haystack, const char* needle);
int mystrcasecmp(const char* str1, const char* str2);
```

```
char* mystrchr(const char* str, int c);
char* mystrrchr(const char* str, int c);
int mystrncmp(const char* str1, const char* str2, size_t n);
char* mystrnstr(const char* haystack, const char* needle, size_t n);
char* mystrncat(char* dest, const char* src, size_t n);
int mystrncasecmp(const char* str1, const char* str2, size_t n);
// a. mystrcpy
char* mystrcpy(char* dest, const char* src) {
 char* ptr = dest;
 while (*src != '\0') {
 *dest = *src;
 dest++;
 src++;
 }
 *dest = '\0';
 return ptr;
}
// b. mystrlen
size_t mystrlen(const char* str) {
 size_t len = 0;
 while (*str != '\0') {
 len++;
 str++;
 }
 return len;
}
// c. mystrcmp
int mystrcmp(const char* str1, const char* str2) {
 while (*str1 && (*str1 == *str2)) {
 str1++;
 str2++;
```

```
 }
 return *(const unsigned char*)str1 - *(const unsigned char*)str2;
}
//d. mystrcat
char* mystrcat(char* dest, const char* src) {
 char* ptr = dest;
 while (*dest != '\0') {
 dest++;
 }
 while (*src != '\0') {
 *dest = *src;
 dest++;
 src++;
 }
 *dest = '\0';
 return ptr;
}
// e. mystrncpy
char* mystrncpy(char* dest, const char* src, size_t n) {
 char* ptr = dest;
 while (n-- && (*src != '\0')) {
 *dest = *src;
 dest++;
 src++;
 }
 *dest = '\0';
 return ptr;
}
// f. mystrupper
char* mystrupper(char* str) {
 char* ptr = str;
```

```
 while (*str != '\0') {
 if (*str >= 'a' && *str <= 'z') {
 *str -= 32;
 }
 str++;
 }
 return ptr;
}
// g. mystrlower
char* mystrlower(char* str) {
 char* ptr = str;
 while (*str != '\0') {
 if (*str >= 'A' && *str <= 'Z') {
 *str += 32;
 }
 str++;
 }
 return ptr;
}
// h. mystrrev
char* mystrrev(char* str) {
 char* start = str;
 char* end = str;
 char temp;
 while (*end != '\0') {
 end++;
 }
 end--;
 while (start < end) {
 temp = *start;
 *start = *end;
```

```
 *end = temp;
 start++;
 end--;
 }
 return str;
}
// i. mystrstr
char* mystrstr(const char* haystack, const char* needle) {
 size_t needle_len = mystrlen(needle);
 while (*haystack != '\0') {
 if (mystrncmp(haystack, needle, needle_len) == 0) {
 return (char*)haystack;
 }
 haystack++;
 }
 return NULL;
}
// j. mystrcasecmp
int mystrcasecmp(const char* str1, const char* str2) {
 while (*str1 && (*str1 == *str2 || (unsigned char)*str1 == (unsigned char)*str2 + 32
|| (unsigned char)*str1 + 32 == (unsigned char)*str2)) {
 str1++;
 str2++;
 }
 return (unsigned char)*str1 - (unsigned char)*str2;
}
// k. mystrchr
char* mystrchr(const char* str, int c) {
 while (*str != '\0') {
 if (*str == c) {
 return (char*)str;
```

```
 }
 str++;
 }
 return NULL;
}
// l. mystrrchr
char* mystrrchr(const char* str, int c) {
 char* last_occurrence = NULL;
 while (*str != '\0') {
 if (*str == c) {
 last_occurrence = (char*)str;
 }
 str++;
 }
 return last_occurrence;
}
// m. mystrncmp
int mystrncmp(const char* str1, const char* str2, size_t n) {
 while (n-- && (*str1 == *str2)) {
 if (*str1 == '\0') {
 return 0;
 }
 str1++;
 str2++;
 }
 return *(const unsigned char*)str1 - *(const unsigned char*)str2;
}
// n. mystrnstr
char* mystrnstr(const char* haystack, const char* needle, size_t n) {
 size_t needle_len = mystrlen(needle);
 while (n-- && *haystack != '\0') {
```

```
 if (mystrncmp(haystack, needle, needle_len) == 0) {
 return (char*)haystack;
 }
 haystack++;
 }
 return NULL;
}
// o. mystrncat
char* mystrncat(char* dest, const char* src, size_t n) {
 char* ptr = dest;
 while (*dest != '\0') {
 dest++;
 }
 while (n-- && (*src != '\0')) {
 *dest = *src;
 dest++;
 src++;
 }
 *dest = '\0';
 return ptr;
}
// p. mystrncasecmp
int mystrncasecmp(const char* str1, const char* str2, size_t n) {
 while (n-- && (*str1 && (*str1 == *str2 || (unsigned char)*str1 == (unsigned
char)*str2 + 32 || (unsigned char)*str1 + 32 == (unsigned char)*str2))) {
 str1++;
 str2++;
 }
 return (unsigned char)*str1 - (unsigned char)*str2;
}
```