1. Write a program to find reverse of given string without using string library function.
2. Write a user defined function to copy one string to another without using library ( ) function.
3. Write user defined functions to implement following string operations
   (i) strcat ii. strlen
4. Write a program to check whether the entered string is a palindrome. (Do not use the string header file)
5. Write a program to count blank spaces, digits, vowels and consonants in the string.
6. Write a program in C to show all possible string library functions.
7. Write a program in C to Remove Duplicates from a String.
8. Write a program in C to Check If Two Strings are Anagrams.
9. Write a program in C to Reverse Words in a String

1. Write a program to find reverse of given string without using string library function.

```
#include <stdio.h>

int main()
{
   char s[1000], r[1000];
   int length = 0, i;

   // Input the string
   printf("Input a string: ");
   scanf("%s", s);

   // Calculate string length manually
   while (s[length] != '\0') {
      length++;
   }

   // Reverse the string
   for (i = 0; i < length; i++) {
      r[i] = s[length - i - 1];
   }
   r[i] = '\0'; // Null-terminate the reversed string

   // Print the reversed string
   printf("Reversed string: %s\n", r);

   return 0;
}
```

2. Write a user defined function to copy one string to another without using library ( ) function.

```
#include <stdio.h>

// User-defined function to copy a string
void stringCopy(char source[], char destination[]) {
   int i = 0;
   while (source[i] != '\0') {
      destination[i] = source[i]; // Copy each character
      i++;
   }
   destination[i] = '\0'; // Null-terminate the destination string
}

int main()
{
   char source[1000], destination[1000];

   // Input the source string
   printf("Enter the source string: ");
   scanf("%s", source);

   // Call the user-defined function to copy the string
   stringCopy(source, destination);

   // Display the copied string
   printf("Copied string: %s\n", destination);

   return 0;
}
```

3. Write user defined functions to implement following string operations
    i. strcat ii. strlen

```
#include <stdio.h>

// Function to calculate the length of a string
int stringLength(char str[]) {
   int length = 0;
   while (str[length] != '\0') {
      length++;
   }
   return length;
}

// Function to concatenate two strings
void stringConcat(char str1[], char str2[], char result[]) {
   int i = 0, j = 0;

   // Copy the first string into result
   while (str1[i] != '\0') {
      result[i] = str1[i];
      i++;
   }

   // Append the second string into result
   while (str2[j] != '\0') {
      result[i] = str2[j];
      i++;
      j++;
   }

   // Null-terminate the concatenated string
   result[i] = '\0';
}

int main()
{
   char str1[1000], str2[1000], result[2000];
   int length;

   // Input for string operations
   printf("Enter the first string: ");
   scanf("%s", str1);
   printf("Enter the second string: ");
```

```
    scanf("%s", str2);


    // Calculate and display the length of the first string
    length = stringLength(str1);
    printf("Length of the first string: %d\n", length);

    // Concatenate the strings and display the result
    stringConcat(str1, str2, result);
    printf("Concatenated string: %s\n", result);

    return 0;
}
```

4. Write a program to check whether the entered string is a palindrome. (Do not use the string header file)

```
#include <stdio.h>

int main()
{
   char str[1000];
   int length = 0, isPalindrome = 1;

   // Input the string
   printf("Enter a string: ");
   scanf("%s", str);

   // Calculate the length of the string
   while (str[length] != '\0') {
      length++;
   }

   // Check if the string is a palindrome
   for (int i = 0; i < length / 2; i++) {
      if (str[i] != str[length - i - 1]) {
         isPalindrome = 0; // Not a palindrome
         break;
      }
   }

   // Output the result
   if (isPalindrome) {
      printf("The string is a palindrome.\n");
   } else {
      printf("The string is not a palindrome.\n");
   }

   return 0;
}
```

5. Write a program to count blank spaces, digits, vowels and consonants in the string.

```
#include <stdio.h>

int main()
{
   char str[1000];
   int spaces = 0, digits = 0, vowels = 0, consonants = 0;

   // Input the string
   printf("Enter a string: ");
   scanf(" %[^\n]", str); // Read the entire line, including spaces

   // Traverse the string
   for (int i = 0; str[i] != '\0'; i++) {
      char ch = str[i];

      // Check for blank spaces
      if (ch == ' ') {
         spaces++;
      }
      // Check for digits
      else if (ch >= '0' && ch <= '9') {
         digits++;
      }
      // Check for vowels
      else if (ch == 'a' || ch == 'e' || ch == 'i' || ch == 'o' || ch == 'u' ||
            ch == 'A' || ch == 'E' || ch == 'I' || ch == 'O' || ch == 'U') {
         vowels++;
      }
      // Check for consonants (letters that are not vowels)
      else if ((ch >= 'a' && ch <= 'z') || (ch >= 'A' && ch <= 'Z')) {
         consonants++;
      }
   }

   // Display the counts
   printf("Blank spaces: %d\n", spaces);
   printf("Digits: %d\n", digits);
   printf("Vowels: %d\n", vowels);
   printf("Consonants: %d\n", consonants);

   return 0;
}
```

6. Write a program in C to show all possible string library functions.

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>

int main() {
   char str[100], str2[100], set[100], ch;
   int n;

   // Basic String Library Functions Demonstration

   // Example 1: strlen()
   printf("Enter a string: ");
   fgets(str, sizeof(str), stdin);
   str[strcspn(str, "\n")] = '\0';  // Remove the trailing newline
   printf("Length of the string: %lu\n", strlen(str));

   // Example 2: strcmp()
   printf("\nEnter another string to compare: ");
   fgets(str2, sizeof(str2), stdin);
   str2[strcspn(str2, "\n")] = '\0';  // Remove the trailing newline
   int result = strcmp(str, str2);
   if (result == 0)
      printf("The two strings are identical.\n");
   else if (result < 0)
      printf("The first string is lexicographically smaller.\n");
   else
      printf("The first string is lexicographically greater.\n");

   // Example 3: strcpy()
   printf("\nEnter a string to copy to another string: ");
   fgets(str2, sizeof(str2), stdin);
   str2[strcspn(str2, "\n")] = '\0';  // Remove the trailing newline
   char str_copy[100];
   strcpy(str_copy, str2);
   printf("String copied to another string: %s\n", str_copy);

   // Example 4: strcat()
   printf("\nEnter another string to concatenate: ");
   fgets(str2, sizeof(str2), stdin);
   str2[strcspn(str2, "\n")] = '\0';  // Remove the trailing newline
   strcat(str, str2);
   printf("Concatenated string: %s\n", str);
```

```
// Example 5: strchr()
printf("\nEnter a character to search for in the string: ");
scanf("%c", &ch);
getchar();  // Consume newline character
char *char_pos = strchr(str, ch);
if (char_pos != NULL)
   printf("First occurrence of '%c' found at position: %ld\n", ch, char_pos - str);
else
   printf("Character '%c' not found in the string.\n", ch);

// Example 6: strstr()
printf("\nEnter a substring to search for: ");
fgets(set, sizeof(set), stdin);
set[strcspn(set, "\n")] = '\0';  // Remove the trailing newline
char *sub_str_pos = strstr(str, set);
if (sub_str_pos != NULL)
   printf("Substring found: %s\n", sub_str_pos);
else
   printf("Substring not found.\n");

// Advanced String Library Functions Demonstration

// Example 7: strncpy()
printf("\nEnter the number of characters to copy from the string: ");
scanf("%d", &n);
getchar();  // To consume the newline character left by scanf
strncpy(str_copy, str, n);
str_copy[n] = '\0';
printf("First %d characters copied: %s\n", n, str_copy);

// Example 8: strncat()
printf("\nEnter the number of characters to concatenate: ");
scanf("%d", &n);
getchar();  // To consume the newline character left by scanf
strncat(str, str2, n);
printf("Concatenated string (first %d characters): %s\n", n, str);

// Example 9: strpbrk()
printf("\nEnter a set of characters to search for in the string: ");
fgets(set, sizeof(set), stdin);
set[strcspn(set, "\n")] = '\0';  // Remove the trailing newline
char *result = strpbrk(str, set);
if (result != NULL)
   printf("First matching character found: %c\n", *result);
else
   printf("No matching characters found.\n");

// Example 10: strrchr()
```

```
    printf("\nEnter a character to find its last occurrence in the string: ");
    scanf("%c", &ch);
    getchar();  // Consume newline
    char *last_occurrence = strrchr(str, ch);
    if (last_occurrence != NULL)
        printf("Last occurrence of '%c' found at position: %ld\n", ch, last_occurrence - str);
    else
        printf("Character '%c' not found in the string.\n", ch);

    // Example 11: strspn()
    printf("\nEnter a set of characters to check the prefix in the string: ");
    fgets(set, sizeof(set), stdin);
    set[strcspn(set, "\n")] = '\0';  // Remove the trailing newline
    size_t prefix_len = strspn(str, set);
    printf("Length of prefix that contains only characters from the set: %zu\n", prefix_len);

    // Example 12: strcspn()
    printf("\nEnter a set of characters to check for exclusions in the string: ");
    fgets(set, sizeof(set), stdin);
    set[strcspn(set, "\n")] = '\0';  // Remove the trailing newline
    size_t exclusion_len = strcspn(str, set);
    printf("Length of prefix that excludes characters from the set: %zu\n", exclusion_len);

    // Example 13: strnset() (Non-standard function, may not work on all compilers)
    printf("\nEnter the character to set the first few characters of the string to: ");
    scanf("%c", &ch);
    getchar();  // Consume newline
    printf("Enter the number of characters to set: ");
    scanf("%d", &n);
    getchar();  // Consume newline
    strnset(str, ch, n);
    printf("Modified string after setting first %d characters to '%c': %s\n", n, ch, str);

    return 0;
}
```

Explanation:

1. **Basic String Functions**:
    - **`strlen()`**: Finds the length of a string.
    - **`strcmp()`**: Compares two strings lexicographically.
    - **`strcpy()`**: Copies one string to another.
    - **`strcat()`**: Concatenates one string to another.
    - **`strchr()`**: Finds the first occurrence of a character in a string.
    - **`strstr()`**: Finds the first occurrence of a substring in a string.
2. **Advanced String Functions**:

- **`strncpy()`**: Copies a specified number of characters from one string to another.
- **`strncat()`**: Concatenates a specified number of characters from one string to another.
- **`strpbrk()`**: Finds the first matching character from a set of characters in the string.
- **`strrchr()`**: Finds the last occurrence of a character in a string.
- **`strspn()`**: Finds the length of the prefix that only contains characters from a set.
- **`strcspn()`**: Finds the length of the prefix that doesn't contain characters from a set.
- **`strnset()`**: Sets the first `n` characters of a string to a specified character (non-standard).

SAMPLE OUTPUT:

Enter a string: Hello World
Length of the string: 11

Enter another string to compare: Hello World
The two strings are identical.

Enter a string to copy to another string: C Programming
String copied to another string: C Programming

Enter another string to concatenate: is fun
Concatenated string: Hello Worldis fun

Enter a character to search for in the string: o
First occurrence of 'o' found at position: 4

Enter a substring to search for: World
Substring found: World

Enter the number of characters to copy from the string: 5
First 5 characters copied: Hello

Enter the number of characters to concatenate: 3
Concatenated string (first 3 characters): Hello Worldis

Enter a set of characters to search for in the string: lo
First matching character found: l

Enter a character to find its last occurrence in the string: o
Last occurrence of 'o' found at position: 7

Enter a set of characters to check the prefix in the string: Heo

Length of prefix that contains only characters from the set: 2

Enter a set of characters to check for exclusions in the string: W
Length of prefix that excludes characters from the set: 1

Enter the character to set the first few characters of the string to: *
Enter the number of characters to set: 3
Modified string after setting first 3 characters to '*': ***o Worldis

7. Write a program in C to Remove Duplicates from a String

```
#include <stdio.h>
#include <string.h>

void removeDuplicates(char str[]) {
   int i, j, k;
   int length = strlen(str);

   // Traverse the string
   for (i = 0; i < length; i++) {
      // If current character is found in remaining part of the string, remove it
      for (j = i + 1; j < length; j++) {
         if (str[i] == str[j]) {
            for (k = j; k < length; k++) {
               str[k] = str[k + 1];
            }
            length--;  // Reduce string length after removal
            j--;  // Decrement j to check the current character again
         }
      }
   }
   printf("String after removing duplicates: %s\n", str);
}

int main() {
   char str[100];

   printf("Enter a string: ");
   fgets(str, sizeof(str), stdin);
   str[strcspn(str, "\n")] = '\0';  // Remove newline character

   removeDuplicates(str);

   return 0;
}
```

8. Write a program in C to Check If Two Strings are Anagrams

```
#include <stdio.h>
#include <string.h>
#include <ctype.h>

int areAnagrams(char str1[], char str2[]) {
   int count[256] = {0};  // Array to store the frequency of characters

   // If lengths are different, return 0 (not anagrams)
   if (strlen(str1) != strlen(str2))
      return 0;

   // Count the frequency of characters in str1 and str2
   for (int i = 0; str1[i] != '\0'; i++) {
      count[tolower(str1[i])]++;
      count[tolower(str2[i])]--;
   }

   // Check if the frequency of all characters is zero
   for (int i = 0; i < 256; i++) {
      if (count[i] != 0)
         return 0;
   }
   return 1;
}

int main() {
   char str1[100], str2[100];

   printf("Enter first string: ");
   fgets(str1, sizeof(str1), stdin);
   str1[strcspn(str1, "\n")] = '\0';  // Remove newline character

   printf("Enter second string: ");
   fgets(str2, sizeof(str2), stdin);
   str2[strcspn(str2, "\n")] = '\0';  // Remove newline character

   if (areAnagrams(str1, str2))
      printf("Yes, the strings are anagrams.\n");
   else
      printf("No, the strings are not anagrams.\n");

   return 0;
}
```

9. Write a program in C to Reverse Words in a String.

```
#include <stdio.h>
#include <string.h>

void reverseWords(char str[]) {
   int start = 0, end = 0, length = strlen(str);

   // Reverse individual words
   for (end = 0; end < length; end++) {
      if (str[end] == ' ' || str[end] == '\0') {
         int temp_end = end - 1;
         while (start < temp_end) {
            char temp = str[start];
            str[start] = str[temp_end];
            str[temp_end] = temp;
            start++;
            temp_end--;
         }
         start = end + 1;
      }
   }

   // Reverse the last word
   int temp_end = end - 1;
   while (start < temp_end) {
      char temp = str[start];
      str[start] = str[temp_end];
      str[temp_end] = temp;
      start++;
      temp_end--;
   }

   printf("String after reversing words: %s\n", str);
}

int main() {
   char str[100];

   printf("Enter a string: ");
   fgets(str, sizeof(str), stdin);
   str[strcspn(str, "\n")] = '\0';  // Remove newline character

   reverseWords(str);

   return 0;
}
```

Sample Output:

Enter a string: Hello World
String after reversing words: olleH dlroW