```
void arr_copyString(char to[],char from[]);
void ptr_copyString(char *to,char *from);
int main()
{
   char A[50];
   char B[20]="Anaswara";

   char op;
   printf("Enter the option:\na-> array implementation\np->pointer implementation\ne->exit\n");
   scanf("%c",&op);

   switch(op)
   {
      case 'a':
      arr_copyString(A,B);
      printf("\n The copied content is %s\n",A);
      break;
      case 'p':
      ptr_copyString(A,B);
      printf("\n The copied content is %s\n",A);
      break;
      case 'e':
      printf("Exiting\n");
      break;
      default:
      printf("Invalid option\n");
   }
}

void arr_copyString(char to[],char from[])
{
   int i;
   printf("Inside array copying\n");
   for(i=0;from[i] != '\0';i++)
   {
      to[i] = from[i];

   }
   to[i] = '\0';

}


void ptr_copyString(char *to, char *from)
{
   char *originalTo = to;
   printf("Inside pointer copying\n");
   for(;*from != '\0';from++,to++)
   {
      *to = *from;

   }
   *to = '\0';

}
```

Problem 1: Palindrome Checker

Problem Statement:

Write a C program to check if a given string is a palindrome. A string is considered a palindrome if it reads the same backward as forward, ignoring case and non-alphanumeric characters. Use functions like strlen(), tolower(), and isalpha().

Example:

Input: "A man, a plan, a canal, Panama"

Output: "Palindrome"

```
#include <stdio.h>
#include <ctype.h>
#include <string.h>
int isPalindrome(const char* str);
int main()
{
   char str[50];
   printf("Enter the string:");
   scanf("%[^\n]",str);
   if (isPalindrome(str))
   {
      printf("Palindrome\n");
   } else
   {
      printf("Not a palindrome\n");
   }

   return 0;

}

int isPalindrome(const char* str)
{
   int left = 0, right = strlen(str) - 1;

   while (left < right)
   {
      while (left < right && !isalnum(str[left]))
      {
         left++;
      }
      while (left < right && !isalnum(str[right]))
      {
         right--;
      }


      if (tolower(str[left]) != tolower(str[right]))
      {
         return 0;
      }

      left++;
```

```
        right--;
    }

    return 1;
}
```

==

Problem 2: Word Frequency Counter

Problem Statement:

Write a program to count the frequency of each word in a given string. Use strtok() to tokenize the string and strcmp() to compare words. Ignore case differences.

Example:

Input: "This is a test. This test is simple."

Output:

```
Word: This, Frequency: 2

Word: is, Frequency: 2

Word: a, Frequency: 1

Word: test, Frequency: 2

Word: simple, Frequency: 1
```

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

int main()
{
    char *word[10] = {NULL};
    int count[10] = {0};
    char str[50];
    char temp[50];

    printf("Input: ");
    scanf(" %[^\n]", str);

    strcpy(temp, str);


    int i = 0, found = 0;
    char *token = strtok(temp, " .,!?");
    while (token != NULL)
    {
        found = 0;
        for (int j = 0; j < i; j++)
        {
            if (strcmp(word[j], token) == 0)
            {
```

```
            count[j]++;
            found = 1;
            break;
          }
        }


        if (!found)
        {
          word[i] = token;
          count[i]++;
          i++;
        }

        token = strtok(NULL, " .,!?");
    }

    for (int j = 0; j < i; j++)
    {
        printf("Word:%s, Frequency: %d\n", word[j], count[j]);
    }

    return 0;
}
```

===

Problem 3: Find and Replace

Problem Statement:

Create a program that replaces all occurrences of a target substring with another substring in a given string. Use strstr() to locate the target substring and strcpy() or strncpy() for modifications.

Example:

Input:

String: "hello world, hello everyone"

Target: "hello"

Replace with: "hi"

Output: "hi world, hi everyone"

```
#include <stdio.h>
#include <string.h>
#include <ctype.h>
int main()
{
    char str[50];
    printf("Enter the string:");
    scanf("%[^\n]",str);
    char substring[30];
    printf("Enter target string to be replaced:");
    scanf("%s",substring);
    char new_substring[30];
    printf("Enter new substring:");
    scanf("%s",new_substring);

    char result[200] = "";
    char *pos = str;
    char *start = str;

    while ((pos = strstr(start, substring)) != NULL)
    {

        strncat(result, start, pos - start);


        strcat(result, new_substring);


        start = pos + strlen(substring);
    }


    strcat(result, start);

    printf("Modified string is: %s\n", result);
    return 0;
}
```

==

Problem 4: Reverse Words in a Sentence

Problem Statement:

Write a program to reverse the words in a given sentence. Use strtok() to extract words and strcat() to rebuild the reversed string.

Example:

Input: "The quick brown fox"

Output: "fox brown quick The"

```
#include <stdio.h>
#include <ctype.h>
#include <string.h>
void reverse_str(char *str);
int main()
{
    char str[50];
    printf("Enter the string:");
    scanf("%[^\n]",str);
    reverse_str(str);
    char *token = strtok(str," ");
    char mod_str[50]="";



    while(token != NULL)
    {
        reverse_str(token);
        strcat(mod_str,token);
        strcat(mod_str," ");
        token = strtok(NULL," ");
    }

    mod_str[strlen(mod_str) - 1] = '\0';
    printf("Reversed string is %s\n",mod_str);
    return 0;
}

void reverse_str(char *str)
{
    int a=0;
    int b= strlen(str) - 1;
    while(a < b)
    {
        char temp = str[a];

        str[a] = str[b];
        str[b] = temp;
        a++;
        b--;
    }

}
```

==

Problem 5: Longest Repeating Substring

Problem Statement:

Write a program to find the longest substring that appears more than once in a given string. Use strncpy() to extract substrings and strcmp() to compare them.

Example:

Input: "banana"

Output: "ana"

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

void findLongest(char *str)
{
    int n = strlen(str);
    int maxLength = 0;
    char longestSub[100];

    for (int len = 1; len < n; len++)
    {
        for (int i = 0; i <= n - len; i++)
        {
            for (int j = i + 1; j <= n - len; j++)
            {
                if (strncmp(str + i, str + j, len) == 0)
                {
                    if (len > maxLength)
                    {
                        maxLength = len;
                        strncpy(longestSub, str + i, len);
                        longestSub[len] = '\0';
                    }
                    break;
                }
            }
        }
    }

    if (maxLength > 0)
    {
        printf("Longest repeated substring: \"%s\"\n", longestSub);
    }
    else
    {
        printf("No repeated substring found.\n");
    }
}

int main()
{
    char str[100];
    printf("Input: ");
    scanf("%s", str);

    findLongest(str);

    return 0;
}
```

==