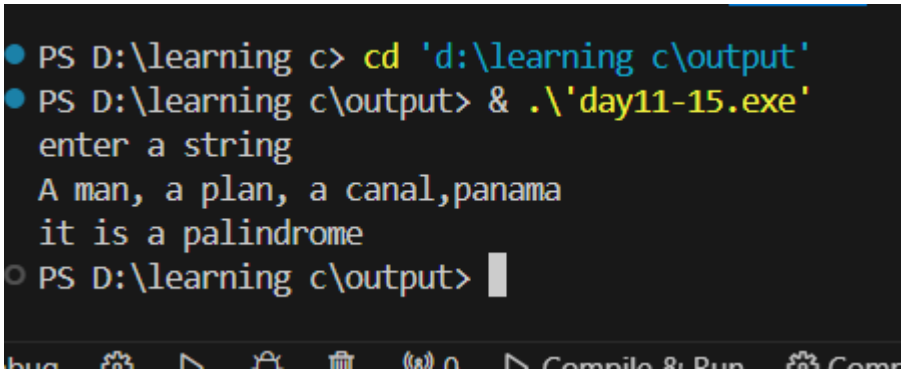
Problem 1: Palindrome Checker
Problem Statement:
Write a C program to check if a given string is a palindrome. A string is considered a palindrome if it reads the same backward as forward, ignoring case and non-alphanumeric characters. Use functions like strlen(), tolower(), and isalpha().
Example:
Input: "A man, a plan, a canal, Panama"
Output: "Palindrome"

```
#include<stdio.h>

#include<string.h>

int main()

{

   char str[20];

   printf("enter a string \n");

   scanf("%s",str);

   int l=strlen(str);

   int start=0;

   int end=l-1;

   int ispali;

   while(start<end)

   {

      while(start<end && !isalpha(str[start]))

      {

         start++;

      }

      while(start<end && !isalpha(str[end]))

      {

         end--;

      }


      if((char)tolower(str[start]==(char)tolower(str[end])))

      {

         ispali=1;
```

```
        }
        else
        {
            ispali=0;
        }
        start++;
        end--;
    }
    if(ispali)
    {
        printf("it is a palindrome");


    }
    else
    {
        printf("not palindrome");
    }



}
```



==========

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 str1[20];

   char words[20][20];

   printf("enter a string(seperate each word by -)");

   scanf("%s",str1);

   char s[2]="-";

   char *token =NULL;

   int frequency[20]={0};

   int wordc=0;

   token=strtok(str1,s);

   while(token!=NULL)

   {

     //printf(" word : %s,frequency: \n",token);


      int found=0;

      for(int i=0;i<wordc;i++)

      {

       if(strcmp(words[i],token)==0)

       {

         frequency[i]++;

         found=1;

         break;

       }
```

```
    }
    if(!found)//if word not found adding to words array
    {
      strcpy(words[wordc],token);
      frequency[wordc]=1;
      wordc++;

    }
     token=strtok(NULL,s);

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

}
```

```
enter a string(seperate each word by -)rinta-mari
a
Word Frequency:
Word: rinta, Frequency: 1
Word: maria, Frequency: 1
PS D:\learning c\output>
```

==========

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>

int main()

{

    char str[20];

    printf("enter a string");

    scanf("%s",str);

    char target[20];

    printf("Target:");

    scanf("%s",target);

    char replace[20];

    printf("Replace with:");

    scanf("%s",replace);

    char word2[50];

    char *pFound=NULL;

    pFound=strstr(str,target);

   // printf("pfound =%s\n",pFound);

    //printf("str =%s\n",str);


    strncpy(word2,str, 5);


    strcat(word2,replace);

    printf("\nThe found word: %s", word2);
```

}

```
PS D:\learning c\output> & .\'day11-18.exe'
enter a stringrintamaria
Target:maria
Replace with:raju

The found word: rintaraju
PS D:\learning c\output>
```

==========

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 <string.h>


int main() {

    char str[50];

    char words[10][50];

    char results[50] = ""; // Initialize the results array

    int wordcount = 0;


    printf("Enter a string: ");

    scanf("%s", str);


    char s[2] = "-";

    char *token = strtok(str, s);
```

```
    // Tokenization and storing in words array
    while (token != NULL) {
        strcpy(words[wordcount], token);
        wordcount++;
        token = strtok(NULL, s);
    }

    // Rebuilding the reversed sentence
    for (int i = wordcount - 1; i >= 0; i--) {
        strcat(results, words[i]);

    }

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

    return 0;
}
```

```
PS D:\learning c\output> & .\'day11-19.exe'
Enter a string: rinta-maria-raju
Reversed sentence: rajumariarinta
PS D:\learning c\output>
```

==========

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>
```

```
int main() {
    char str[10], result[10] = "";
    int maxLength = 0;

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

    int n = strlen(str);

    for (int i = 0; i < n; i++) {
        for (int j = i + 1; j < n; j++) {
            int length = 0;

            while (i + length < n && j + length < n && str[i + length] == str[j + length]) {
                length++;
            }



            if (length > maxLength) {
                maxLength = length;
                strncpy(result, str + i, length);
                result[length] = '\0';
            }
        }
    }

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

```
  }

  return 0;
}
```

```
PS D:\learning c\output> & .\'day11
Enter a string: banana
Longest repeating substring: ana
PS D:\learning c\output>
```

==========

Class work

1. //string functions -strlen

```
#include<stdio.h>
#include<string.h>
int main()
{
  char str1[]="rinta maria";
  printf("length of string is %ld \n",strlen(str1)); //ld cz long integer
}
```

```
PS D:\learning c\output> & .\'day11-1.exe'
length of string is 11
```

2. //copying strings

```
//2 parameters
//para 1-destination.para1-source
#include<stdio.h>
#include<string.h>
int main()
{
 char src[10],dest[10];
```

```
strcpy(dest, "rinta");
strcpy(src, "maria");
printf("%s \t %s",src,dest);
}
```

```
PS D:\learning c\output> & .\'day11-2.exe'
maria    rinta
PS D:\learning c\output>
```

3. //strncpy

```
//3 arguments
//arg1-dest,arg2-src,arg4-max no.of characters
#include<stdio.h>
#include<string.h>
int main()
{
   char str1[10];
   char str2[10];
   strcpy(str1,"maria");
   strncpy(str2,str1,10);
   printf("str1[]=%s \n str2[]=%s",str1,str2);
   return 0;
}
```

```
PS D:\learning c\output> & .\'day11-3.exe'
str1[]=maria
 str2[]=maria
PS D:\learning c\output>
```

4. //concatenate

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

```
int main()
{
   char str1[20];
   char str2[10];
   strcpy(str1,"maria");
   strncpy(str2,str1,10);
   printf("str1[]=%s \n str2[]=%s \n",str1,str2);
   strcat(str1,str2);
   printf("str1[]=%s \n str2[]=%s \n",str1,str2);

   return 0;
}
```

```
PS D:\learning c\output> & .\'day11-4.exe'
str1[]=maria
 str2[]=maria
str1[]=mariamaria
 str2[]=maria
PS D:\learning c\output>
```

5. //comparing strings

```
//use string as argument not character
//if equal returns zero else non zero
//<0 then str1<str2
//>0 then str1>str2
#include<stdio.h>
#include<string.h>
int main()
{
  printf("strcmp(\"A\",\"A\")is");
  printf("%d \n",strcmp("A","A"));
  printf("strcmp(\"A\",\"B\")is");//A ascii is 65 B is 66 ie A<B so -1
```

```
  printf("%d \n",strcmp("C","B"));
  printf("strcmp(\"C\",\"B\")is");
  printf("%d \n",strcmp("C","B"));
  printf("strcmp(\"Apples\",\"apples\")is");
  printf("%d \n",strcmp("Apples","apples"));
  printf("strcmp(\"apples\",\"apple\")is");//compares each character
  printf("%d \n",strcmp("apples","apple"));
  printf("strcmp(\"ABCD\",\"ABBD\")is");//once a diff is found and it stops comparing
  printf("%d \n",strcmp("ABCD","ABBD"));
  printf("strcmp(\"ABBA\",\"AABA\")is");
  printf("%d \n",strcmp("ABBA","AABA"));
  printf("strcmp(\"AABA\",\"ABBA\")is");
  printf("%d \n",strcmp("AABA","ABBA"));

}
```

```
PS D:\learning c\output> & .\'day11-5.exe'
strcmp("A","A")is0
strcmp("A","B")is1
strcmp("C","B")is1
strcmp("Apples","apples")is-1
strcmp("apples","apple")is1
strcmp("ABCD","ABBD")is1
strcmp("ABBA","AABA")is1
strcmp("AABA","ABBA")is-1
PS D:\learning c\output>
```

6.
```
#include<stdio.h>
#include<string.h>
int main()
{
  printf("strcmp(\"Astounding\",\"Astro\")is");
```

```
  printf("%d \n",strcmp("Astounding","Astro"));
  printf("strncmp(\"Astounding\",\"Astro\")is");
  printf("%d \n",strncmp("Astounding","Astro",5));

}
```

```
PS D:\learning c\output> & .\'day11-6.exe'
strcmp("Astounding","Astro")is-1
strncmp("Astounding","Astro")is-3
PS D:\learning c\output>
```

7. //strchr

```
//arg1-string to be searched
//arg2-character u are looking for
//return , when found :address of this position in memory(pointer to the character)
//when not found , returns NULL
#include<stdio.h>
#include<string.h>
int main()
{
   char str[]="the quick brown fox";//string to be searched
   char ch='q';//character we are looking for
   char *pgot_char=NULL;//pointer intilaized to null
   pgot_char=strchr(str,ch);//stores address where ch is found
}
```

8. #include<stdio.h>

```
#include<string.h>
int main()
{
   char str[]="hi iam nrinta";
   int l=strlen(str);
   for(int i=0;i<l;i++)
```

```
    {
        printf("str[%d]=%c ,adress=%p \n",i,str[i],(str+i));
    }
    char ch='n';
    char *pFound=NULL;
    pFound=strchr(str,ch);
    printf("pfound =%c \n",*pFound);
    printf("pfound =%p",pFound);//first occurence of n adress

}
```

```
PS D:\learning c\output> & .\'day11-8.exe'
str[0]=h ,adress=0061FF02
str[1]=i ,adress=0061FF03
str[2]=  ,adress=0061FF04
str[3]=i ,adress=0061FF05
str[4]=a ,adress=0061FF06
str[5]=m ,adress=0061FF07
str[6]=  ,adress=0061FF08
str[7]=n ,adress=0061FF09
str[8]=r ,adress=0061FF0A
str[9]=i ,adress=0061FF0B
str[10]=n ,adress=0061FF0C
str[11]=t ,adress=0061FF0D
str[12]=a ,adress=0061FF0E
pfound =n
pfound =0061FF09
PS D:\learning c\output>
```

9. //searching for substring:strstr()

//returns if found address first occurence of substring

//arg1-string to be searched

//arg2-substring

//case sensitive

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

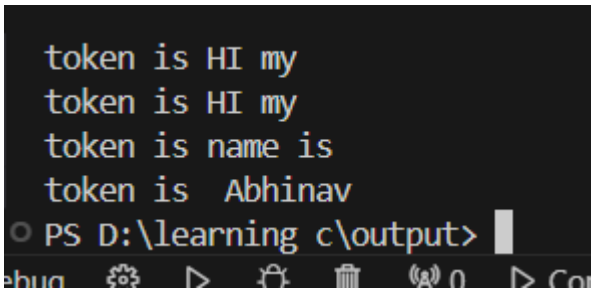
```
#include<string.h>
int main()
{
    char text[]="every dog has his day";
    int l=strlen(text);
    for(int i=0;i<l;i++)
    {
        printf("text[%d]=%c ,adress=%p \n",i,text[i],(text+i));
    }
    char word[]="dog";
    char word2[50];
    char *pFound=NULL;
    pFound=strstr(text,word);
    strncpy(word2,pFound, 3);



    printf("pfound =%c \n",*pFound);
    printf("pfound =%p \n",pFound);
    printf("pfound =%s\n",pFound);
    printf("\nThe found word: %s", word2);




}
```

```
text[1]=v ,adress=0061FEFF
text[2]=e ,adress=0061FF00
text[3]=r ,adress=0061FF01
text[4]=y ,adress=0061FF02
text[5]=  ,adress=0061FF03
text[6]=d ,adress=0061FF04
text[7]=o ,adress=0061FF05
text[8]=g ,adress=0061FF06
text[9]=  ,adress=0061FF07
text[10]=h ,adress=0061FF08
text[11]=a ,adress=0061FF09
text[12]=s ,adress=0061FF0A
text[13]=  ,adress=0061FF0B
text[14]=h ,adress=0061FF0C
text[15]=i ,adress=0061FF0D
text[16]=s ,adress=0061FF0E
text[17]=  ,adress=0061FF0F
text[18]=d ,adress=0061FF10
text[19]=a ,adress=0061FF11
text[20]=y ,adress=0061FF12
pfound =d
pfound =0061FF04
pfound =dog has his day
```

10. #include<stdio.h>

```
#include<string.h>
int main()
{
   char str[]="HI my -name is - Abhinav";
   char s[2]="-";
   char *token =NULL;
   token=strtok(str,s);
   printf("token is %s \n",token);
   while(token!=NULL)
   {
      printf("token is %s \n",token);
      token=strtok(NULL,s);
```

```
    }
}
```



11.
```
#include<stdio.h>
#include<string.h>
#include<ctype.h>
int main()
{
    char buff[100];//input buffer
    int nletters=0;
    int ndigits=0;
    int npunct=0;
    printf("enter a string less than %d characters",100);
    scanf("%s",buff);//read a string to the buffer
    int i=0;//buffer index
    while(buff[i])
    {
      if(isalpha(buff[i]))
      {
         ++nletters;
      }

       else if(isdigit(buff[i]))
       {
            ++ndigits;
       }
```

```
        else if(ispunct(buff[i]))
        {
            ++npunct;

        }
         ++i;

    }
    printf("\n your string as %d letters,%d digits ,%d punctuation \n",nletters,ndigits,npunct);

}
```

```
enter a string less than 100 charactershello123

 your string as 5 letters,3 digits ,0 punctuation
PS D:\learning c\output>
```

12. //converting case

```
//eg :toupper(abcAB) o/p:ABCAB
//typecast to char cz toupper()returns type int
#include <stdio.h>
#include <string.h>
#include <ctype.h> // Include ctype.h for character type functions

int main() {
    char text[100];
    char substring[40];

    printf("Enter the string to be searched: \n");
    scanf("%99s", text);
    printf("Enter the string to be sorted: \n");
    scanf("%39s", substring);
```

```
    printf("String to search: %s\n", text);
    printf("String to sort: %s\n", substring);

    // Convert text to uppercase
    for (int i = 0; (text[i] = (char)toupper(text[i])) != '\0'; ++i) ;




    // Convert substring to uppercase
    for (int i = 0; (substring[i] = (char)toupper(substring[i])) != '\0'; ++i) ;



   // printf("%s",text);


    printf("The second string %s found in first\n", ((strstr(text, substring) == NULL) ? "was not" :
"was"));


    return 0;
}
```

14. //pointer to array

```
//array parameter vs char parameter
#include<stdio.h>
#include<string.h>
int main()
{
    char A[20];
    char B[20]="rinta";
    char choice;
    printf("Choose the method to copy the string:\n");
```

```
    printf("A. Array notation\n");
    printf("B. Pointer notation\n");
    printf("Enter your choice: ");
    scanf("%c", &choice);
    switch (choice)
    {
       case 'A':
         copystring(A, B);
          break;
       case 'B':
           copyString(A, B);
           break;
           default:
           printf("Invalid choice.\n");
            break;

    }

}
// A -array notation
//P-pointer notation
void copystring(char A[],char B[] )
{
    int i;
    for(i=0;B[i]!='\0';++i)
    {
       A[i]=B[i];

    }
     A[i]='\0';
    printf("%s \n",A);
```

```
}

void copyString(char *A,char *B)
{
   char *a=A;
   while(*B!='\0')
   {
     *A=*B;
     ++A;
     ++B;
   }
   *A='\0';
   printf("%s",a);
}
```

```
Choose the method to copy the string:
A. Array notation
B. Pointer notation
Enter your choice: A
rinta
PS D:\learning c\output> cd 'd:\learning c\output'
PS D:\learning c\output> & .\'day11-14.exe'
Choose the method to copy the string:
A. Array notation
B. Pointer notation
Enter your choice: B
rinta
```

15.
```
#include<stdio.h>
#include<string.h>
#include<stdlib.h>
int main()
{
```

```
int *ptr;
int num,i;
//num=(char*)malloc(15);
printf("enter number of elements");
scanf("%d",&num);
printf("\n");
printf("the number entered is n=%d \n",num);
//dynamically allocate memory for array
ptr=(int *)malloc(num*sizeof(int));
//check whether the memory is allocated sucessfully or not
if(ptr==NULL)
{
   printf("memory not allocated \n");
   exit(0);//to terminate the program or use return 0;
}
else
{
   printf("memory allocated \n");
}
//populating the array

for(i=0;i<num;i++)
{
   ptr[i]=i+1;

}
//displaying the array
for(i=0;i<num;i++)
{
   printf("%d",ptr[i]);
}
```

```
free(ptr);//releasing or free the pointer


}
```

```
PS D:\learning c\output> & .\'day11-17.exe'
enter number of elements3

the number entered is n=3
memory allocated
123
```

```
PS D:\learning c\output> cd 'd:\learning c\output'
PS D:\learning c\output> & .\'day11-12.exe'
Enter the string to be searched:
riNTA
Enter the string to be sorted:
rINta
String to search: riNTA
String to sort: rINta
The second string was found in first
PS D:\learning c\output>
```