STRING

```
#include <stdio.h>
#include<string.h>
int main()
{
   char str1[10];
   char str2[10];
   strcpy(str1,"antony");
   strncpy(str2, str1, 3);
   printf("str1[]=%s \t str2[]=%s",str1,str2);
   return 0;
}
```

```
strcpy(dest,sourc)
strncpy(dest,source,no of chara from source);
strcat(dest,source)
strcmp(str1,str2)
strncmp(str1,str2,no of characters to be compared)
```

----

strchr- character search in string

```
Char str[]="my name is antony";
Char ch='i';
char *p_str=NULL;   → declare a pointer to srch a character
p_str=strchr(str,ch);
```

```
#include <stdio.h>
#include<string.h>
int main()
{
  char str[]="my name is antony";
  int l=strlen(str);
  for(int i=0;i<l;i++){
     printf("str[%d]=%c ,address =%p\n",i,str[i],str+i);
  }
  char c='a';
  char *p_str=NULL;
  p_str=strchr(str,c);
  printf("p_str=%p",p_str);
}

//returns the first occurrence of the character
//pointer is used to store the address of the first occurrence
```

```
str[0]=m ,address =0x7ffc58f79f70
str[1]=y ,address =0x7ffc58f79f71
str[2]=  ,address =0x7ffc58f79f72
str[3]=n ,address =0x7ffc58f79f73
str[4]=a ,address =0x7ffc58f79f74
str[5]=m ,address =0x7ffc58f79f75
str[6]=e ,address =0x7ffc58f79f76
str[7]=  ,address =0x7ffc58f79f77
str[8]=i ,address =0x7ffc58f79f78
str[9]=s ,address =0x7ffc58f79f79
str[10]=  ,address =0x7ffc58f79f7a
str[11]=a ,address =0x7ffc58f79f7b
str[12]=n ,address =0x7ffc58f79f7c
str[13]=t ,address =0x7ffc58f79f7d
str[14]=o ,address =0x7ffc58f79f7e
str[15]=n ,address =0x7ffc58f79f7f
str[16]=y ,address =0x7ffc58f79f80
p_str=0x7ffc58f79f74

...Program finished with exit code 0
Press ENTER to exit console.
```

----

strstr-search substring

```
Char str[]="my name is antony";
Char word[]="name";
char *p_str=NULL;          declare a pointer to srch a word
p_str=strstr(str,word);    this is case sensitive
```

```
#include <stdio.h>
#include<string.h>
int main()
{
  char str[]="my name is antony";
  int l=strlen(str);
  for(int i=0;i<l;i++){
     printf("str[%d]=%c ,address =%p\n",i,str[i],str+i);
  }
   char word[]="ant";
   char *p_str=NULL;
   p_str=strstr(str,word);

   printf("%s found from  p_str=%p",word,p_str);
}
```

```
str[0]=m ,address =0x7fff6ec13860
str[1]=y ,address =0x7fff6ec13861
str[2]=  ,address =0x7fff6ec13862
str[3]=n ,address =0x7fff6ec13863
str[4]=a ,address =0x7fff6ec13864
str[5]=m ,address =0x7fff6ec13865
str[6]=e ,address =0x7fff6ec13866
str[7]=  ,address =0x7fff6ec13867
str[8]=i ,address =0x7fff6ec13868
str[9]=s ,address =0x7fff6ec13869
str[10]=  ,address =0x7fff6ec1386a
str[11]=a ,address =0x7fff6ec1386b
str[12]=n ,address =0x7fff6ec1386c
str[13]=t ,address =0x7fff6ec1386d
str[14]=o ,address =0x7fff6ec1386e
str[15]=n ,address =0x7fff6ec1386f
str[16]=y ,address =0x7fff6ec13870
ant found from  p_str=0x7fff6ec1386b
```

Strtok-tokenization
Sub dividing a string based on some delimiters

```
#include <stdio.h>
#include<string.h>
int main()
{
  char str[]="my- name -is- antony";
//  int l=strlen(str);
//  for(int i=0;i<l;i++){
//     printf("str[%d]=%c ,address =%p\n",i,str[i],str+i);
//  }
   char token[2]="-";
   char *p_token=NULL;
   p_token=strtok(str,token);

   printf("token=%s",p_token);
}
```

```
token=my

//prints the 1st token when 1st
delimiter occur
```

****

```
#include <stdio.h>
#include<string.h>
int main()
{
  char str[]="my- name -is- antony";
//  int l=strlen(str);
//  for(int i=0;i<l;i++){
//     printf("str[%d]=%c ,address =%p\n",i,str[i],str+i);
//  }
   char token[2]="-";
   char *p_token=NULL;
   p_token=strtok(str,token);
   while(p_token!=NULL){
     printf("token=%s\n",p_token);
     p_token=strtok(NULL,token);
   }
   return 0;
}
```

```
token=my
token= name
token=is
token= antony
```

- First it will check for the "-" then stops when found and stores in pointer(p_token)
- Thwn to print the next token we are checking again bys starting looking for **NULL** as starting
- This is because the first token will end by giving a null at end of token q string

----

STRING ANALYSIS

```
#include <stdio.h>
#include<string.h>
#include<ctype.h>
int main()
{
   char buff[100];
   int nletters=0;
   int ndigits=0;
   int npunct;
   printf("enter a string of less than %d characters \n",100);
   scanf("%s",buff);
   int i=0;
   while(buff[i]){
      if(isalpha(buff[i])){
         ++nletters;
      }
      else if(isdigit(buff[i])){
         ++ndigits;
      }
      else if(ispunct(buff[i])){
         ++npunct;
      }
      ++i;
   }
   printf("\n string contained %d letters\n %d digits\n %d punctuation\n",nletters,ndigits,npunct);
   return 0;
}
```

```
enter a string of less than 100 characters
Hliam*123.sd

 string contained 7 letters
 3 digits
 2 punctuation
```

```
#include <stdio.h>
#include<string.h>
#include <ctype.h>
int main()
{
   char text1[100];
   char text2[40];
   printf("enter a string :\n");
   scanf(" %[^\n]",text1);
   printf("enter the word to search:\n");
   scanf(" %[^\n]",text2);
   for(int i=0;(text1[i]=(char)toupper(text1[i]))!='\0';++i);//convert to upper case;
   for(int i=0;(text2[i]=(char)toupper(text2[i]))!='\0';++i);//
   printf("%s\n",text1);
   printf("%s\n",text2);
   printf("the second string %s found in the first\n",((strstr(text1,text2)==NULL)?"was not":"was"));
   return 0;
}
```

```
ant
HI IN AM ANTONY
ANT
the second string was found in the first
```

type casting is done since it retuen the integer value(ASCI) of each letter;

POINTERS IN STRINGS

To copy a string to another using ptr and normal string operation

```
#include <stdio.h>
#include<string.h>
#include <ctype.h>
void copyString(char to[],char from[]);
void p_copyString(char *to,char *from);
int main()
{
  char text1[20]="antony";
  char text2[20];
  char op;
  printf("select 'c' or 'p'\n");
  scanf("%c",&op);
  switch(op){
    case 'c':
      copyString(text2,text1);
      break;
    case 'p':
      p_copyString(text2,text1);
      break;
    default:
      printf("invalid operation\n");
  }
  return 0;
}
void copyString(char to[],char from[]){
  int i;
  for(i=0;from[i]!='\0';i++){
    to[i]=from[i];
  }
  to[i]='\0';
  printf("%s",to);
}
void p_copyString(char *to, char *from){
  char *start=to;
  while(*from!='\0'){
    *to=*from;
    *from++;
    *to++;
  }
  *to='\0';

   printf("%s",start);
}
```

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>
#include <ctype.h>
void copyString(char to[],char from[]);
void p_copyString(char *to,char *from);
int main()
{
  int pcount=0;
  char text1[40]="A man, a plan, a canal, Panama";

  int len=strlen(text1);
  int j=len-1;
  for(int k =0;(text1[k]=(char)tolower(text1[k]))!='\0';++k);
  printf("%s\n",text1);
  for(int i=0;i<j;){
    if(!isalpha(text1[i])){
      i++;
      continue;
    }
    if(!isalpha(text1[j])){
      j--;
      continue;
    }

    if(text1[i]!=text1[j]){
      printf("not palinfrome\n");
      return 0;
    }
    i++;
    j--;
  }
  printf("palindrome");
  return 0;
}
```

==
======

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>
#include <ctype.h>
void toLowerCase(char *str);
int main() {
  char input[200] = "this is a test. this test is simple.";
  int frequency[40];
  int wordCount = 0;
  char words[20][20];
```

```
    char *token = strtok(input, " .,!?");
    while (token != NULL) {
        int found = 0;
        for (int i = 0; i < wordCount; i++) {
            if (strcmp(words[i], token) == 0) {
                frequency[i]++;
                found = 1;
                break;
            }
        }
        if (0==found) {
            strcpy(words[wordCount], token);
            frequency[wordCount] = 1;
            wordCount++;
        }
        token = strtok(NULL, " .,!?");
    }
   for (int i = 0; i < wordCount; i++) {
        printf("Word: %s, Frequency: %d\n", words[i], frequency[i]);
    }

    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>
#include<stdlib.h>

int main() {
   char string[]="abc hello world, hello everyone";
   char target[]="hello";
   char temp[100];
   int string_len=strlen(string);
   int target_len=strlen(target);
   char *p;
   char *str=string;
   char rep[]="hi";
   int rep_len=strlen(rep);
   int index=0;
   while((p=strstr(str,target))!=NULL){
       strncpy(temp+index,str,p-str);
       index=index+(p-str);
       strcpy(temp+index,rep);
       index=index+rep_len;
```

```
    str=p+target_len;
  }
  strcpy(temp + index, str);
  temp[index + strlen(str)] = '\0';
  printf("%s\n", temp);
  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 <string.h>
#include <ctype.h>
#include<stdlib.h>

int main() {
  char string[]="The quick brown fox";
  char rev_string[100]="";
  char temp_string[100]="";
  char *words[100];
  int word_count=0;
  char *ptr;
  ptr=strtok(string," ");
  while(ptr!=NULL){
    words[word_count++]=ptr;
    ptr=strtok(NULL," ");

  }
  for(int i=word_count-1;i>=0;i--){
    strcat(rev_string,words[i]);
    if(i>0){
      strcat(rev_string," ");
    }

  }
  printf("%s",rev_string);

  return 0;
```

==
======

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 longest(char *str) {
  int n = strlen(str);
  int maxLen = 0;
  char string[100] = "";


  for (int len = 1; len <= n / 2; len++) {
    for (int i = 0; i <= n - len; i++) {
      char substring[100];
      strncpy(substring, str + i, len);
      substring[len] = '\0';


      for (int j = i + 1; j <= n - len; j++) {
        char otherSubstring[100];
        strncpy(otherSubstring, str + j, len);
        otherSubstring[len] = '\0';


        if (strcmp(substring, otherSubstring) == 0 && len > maxLen) {
          maxLen = len;
          strcpy(string, substring);
        }
      }
    }
  }

  if (maxLen > 0) {
    printf("Longest Repeating Substring: %s\n", string);
  } else {
    printf("No repeating substring found.\n");
  }
}

int main() {
  char str[] = "banana";
  longest(str);
  return 0;
}
```

==
======

Dynamic memory allocation (mmry allocated in heap)
<stdlib.h>

1. Malloc
2. Calloc
3. Realloc

1.Malloc

```
Int *pnum=(int*)malloc(100);     // 100 bytes of memory is allocated
                                 //starting position address is stored in the pnum
```

- If we don't know size of int then
- Int *pnum=(int*)malloc(25*sizeof(int));

- Type casting is used bcoz *pnum can only store address not an integr value;

- When malloc is is used it returns a address but compiler consider it as a hex number so a datatype mismatch occurs.

```
free(pnum);
pnum=NULL;
#include <stdio.h>
#include <string.h>
#include <ctype.h>
#include<stdlib.h>

int main() {
  // char *str;
  // str=(char*)malloc(15);
  // strcpy(str,"antony");
  // printf("string=%s,address=%p",str,str);
  int *ptr;
  int num,i;
  printf("enter number of elemnts");
  scanf("%d",&num);
  printf("\n");
  printf("the number entered is n=%d\n",num);
  // dynamic allocation
  ptr=(int*)malloc(num*sizeof(int));
  // check for dynamic allocation
  if(NULL==ptr){
    printf("mmemory not allocated\n");
    exit(0);// or return 0;
  }
  else{
    printf("memory is allocated succesfully\n");
  }
  //to enter elements dynamically
  printf("enter the elements\n");
  for(int i=0;i<num;i++){
    ptr[i+1]=i+1;
  }
  for(int i=0;i<num;i++){
    printf("%d",ptr[i+1]);
  }
  free(ptr);
  return 0;
}
```

```
enter number of elemnts5

the number entered is n=5
memory is allocated succesfully
enter the elements
12345
```
