PHASE 1 DAY 11

STRING FUNCTIONS

STRCAT

```
#include <stdio.h>
#include<string.h>
int main()
{
char str1[20]="Abhinav";
char str2[20]="karan";
printf("001str1=%s str2=%s\n",str1,str2);

strcat(str1,str2);

printf("002str1=%s str2=%s\n",str1,str2);

   return 0;
}
```

STRCMP

```
#include <stdio.h>
#include<string.h>
int main()
{
   /*
   strcmp

   str1>str2 >0 (1)
   str1<str2 <0 (-1)
   str1=str2  0 (0)
  It checks for each character when it detects the first change.
   Ascii values are compared.
 Other differences are not checked.
   */

printf("strcmp(\"A\",\"A\")is");
printf(" %d\n",strcmp("A","A"));

printf("strcmp(\"A\",\"B\")is");
printf(" %d\n",strcmp("A","B"));
```

```
printf("strcmp(\"B\",\"A\")is");
printf(" %d\n",strcmp("B","A"));

printf("strcmp(\"C\",\"A\")is");
printf(" %d\n",strcmp("C","A"));

printf("strcmp(\"A\",\"Z\")is");
printf(" %d\n",strcmp("A","Z"));

printf("strcmp(\"apples\",\"apple\")is");
printf(" %d\n",strcmp("apples","apple"));

printf("strcmp(\"D\",\"A\")is");
printf(" %d\n",strcmp("D","A"));

printf("strcmp(\"ABCD\",\"ABBD\")is ");
printf(" %d\n",strcmp("ABCD","ABBD"));

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

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


 return 0;
}
```

Output:

```
strcmp("A","A")is 0
strcmp("B","A")is 1
strcmp("C","A")is 1
strcmp("A","Z")is -1
strcmp("apples","apple")is 1
strcmp("D","A")is 1
strcmp("ABCD","ABBD")is  1
strcmp("Astounding","Astro")is -1
strncmp("Astounding","astro")is -3
```

STRCHR

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


   char str[]="hi my name is Abhinav";
   char ch='n';
   int len=strlen(str);

   for(int i=0;i<len;i++)
   {
      printf("str[%d]->%c,address->%p\n",i,str[i],(str+i));
   }
   char *pFound=NULL;
   pFound=strchr(str,ch);
   printf("pFound=%p",pFound);
   return 0;
}


/*
output:
str[0]->h,address->0x7fffca1ea4a0
str[1]->i,address->0x7fffca1ea4a1
str[2]-> ,address->0x7fffca1ea4a2
str[3]->m,address->0x7fffca1ea4a3
str[4]->y,address->0x7fffca1ea4a4
str[5]-> ,address->0x7fffca1ea4a5
str[6]->n,address->0x7fffca1ea4a6
str[7]->a,address->0x7fffca1ea4a7
str[8]->m,address->0x7fffca1ea4a8
str[9]->e,address->0x7fffca1ea4a9
str[10]-> ,address->0x7fffca1ea4aa
str[11]->i,address->0x7fffca1ea4ab
str[12]->s,address->0x7fffca1ea4ac
str[13]-> ,address->0x7fffca1ea4ad
str[14]->A,address->0x7fffca1ea4ae
str[15]->b,address->0x7fffca1ea4af
str[16]->h,address->0x7fffca1ea4b0
```

```
str[17]->i,address->0x7fffca1ea4b1
str[18]->n,address->0x7fffca1ea4b2
str[19]->a,address->0x7fffca1ea4b3
str[20]->v,address->0x7fffca1ea4b4
pFound=0x7fffca1ea4a6
*/
```

STRSTR

```
#include <stdio.h>
#include<string.h>
int main()
{
   char str[]="Every dog has a day";
   char str1[]="dog";
   int len=strlen(str1);

   char *pFound=NULL;
   pFound=strstr(str,str1);
   printf("the word :");
    for(int i=0;i<len;i++)
   {
   printf( "%c",pFound[i]);
   }

   printf("\nthe word %s \n",pFound);
   printf("the word %s is found in pFound=%p",str1,pFound);
   return 0;
}


/*
output
the word :dog
the word dog has a day
the word dog is found in pFound=0x7ffef8f57c96
*/
```

TOKENIZATION

STRTOK

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

   char str[]="hi my-name is-abhinav";
   char str1[]="-";

   char s[2]="-";

   char *token=NULL;
   token=strtok(str,str1);

   printf("token= %s \n",token);//Prints upto first delimiter

  while(token!=NULL)//All other parts in between delimiter
  {
     printf("token =%s\n",token);
     token=strtok(NULL,str1);
  }
   return 0;
}

/*
***output***
token= hi my
token =hi my
token =name is
token =abhinav

*/
```

ISALNUM , ISDIGIT ,ISPUNCT

```
#include <stdio.h>
#include<string.h>
#include<ctype.h>
int main()
{
char buf[100];
int nLetters=0;
int nDigits=0;
int nPunct=0;

printf("enter string: ");
scanf("%s",buf);

int i=0;

while(buf[i])
{
   if(isalpha(buf[i]))
   ++nLetters;
   else if(isdigit(buf[i]))
   ++nDigits;
   else if(ispunct(buf[i]))
   ++nPunct;
   ++i;
}

printf("\n your string contained %d letters,%d digits and %d punctuations
characters.\n",nLetters,nDigits,nPunct);
   return 0;
}

/*  output
enter string: nanda123!@!
your string contained 5 letters,3 digits and 3 punctuations characters.
*/
```

TOUPPER TOLOWER

```
#include <stdio.h>
#include<string.h>
#include<ctype.h>
int main()
{
char text[100];
char substring[40];

printf("enter string to be searched: ");
scanf("%s",text);

printf("enter string sought: ");
scanf("%s",substring);

printf("\nFirst string : %s",text);
printf("\nSecond string : %s",substring);

   for(int i=0;(text[i]=(char)toupper(text[i]))!='\0';i++);
    for(int i=0;(substring[i]=(char)toupper(substring[i]))!='\0';i++);

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

   return 0;
}
```

Assignments

1.copy string

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

void copyString_pointer(char *,char *);
void copyString_array(char [],char []);

int main()
{
        char a[20];
        char b[20];
```

```
        char choice;
        printf("Enter two strings\n");
        printf("\nEnter the first string:");
        scanf("%[^\n]",a);
        getchar();
        printf("\nEnter the Secong string:");
        scanf("%[^\n]",b);
    getchar();
        printf("\nEnter choice (p-pointer/a-array):");
        scanf(" %c",&choice);

        switch(choice)
        {
        case 'p':
        {
                copyString_pointer(a,b);
                break;
        }
        case 'a':
        {
                copyString_array(a,b);
                break;
        }
        default:
        printf("\nInvalid option");

        }
        return 0;
}

void copyString_array(char to[], char from[])
{
    int i;
    for (i = 0; from[i] != '\0'; ++i)
    {
        to[i] = from[i];
    }
    to[i] = '\0';
        printf("\nCopied using array notation:%s\n",to);
```

```
}

void copyString_pointer(char *to, char *from)
{
   char *start = to;
   for (; *from != '\0'; ++from, ++to)
   {
      *to = *from;
   }
   *to = '\0';
      printf("\nCopied using pointer notation:%s\n",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"
Answer:

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

int isPalindrome(char str[]);

int main()
{
   char str[100];
   printf("Enter a string: ");
   scanf("%[^\n]",str);


   if (isPalindrome(str))
      printf("Palindrome\n");
   else
```

```
        printf("Not a palindrome\n");
    return 0;
}

int isPalindrome(char str[])
{
    int i = 0, j = strlen(str) - 1;
    while (i < j)
    {
        while (i < j && !isalnum(str[i])) i++;
        while (i < j && !isalnum(str[j])) j--;

        if (tolower(str[i]) != tolower(str[j]))
        return 0;

        i++;
        j--;
    }
    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>
#include <ctype.h>

int main()
```

```
{
    char str[200], words[100][50];
    int frequency[100] = {0}, count = 0;
    printf("Enter a string: ");
    scanf("%[^\n]",str);

    tolower(str);

    char *token = strtok(str, " .,!?;");
    while (token) {
        int found = 0;
        for (int i = 0; i < count; i++)
        {
            if (strcmp(words[i], token) == 0)
            {
                frequency[i]++;
                found = 1;
                break;
            }
        }
        if (!found)
        {
            strcpy(words[count], token);
            frequency[count++] = 1;
        }
        token = strtok(NULL, " .,!?;");
    }

    for (int i = 0; i < count; 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"
Answer:

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

void findandreplace(char str[], char target[], char replacement[])
{
   char result[500] = "";
   char *pos = str, *found;

   while ((found = strstr(pos, target)) != NULL)
   {
      int new=found-pos;
      strncat(result,pos,found-pos); //to concatenate from 2 strings and number of characters is the
3rd argument
      strcat(result, replacement);
      pos = found + strlen(target);
   }
   strcat(result, pos);
   strcpy(str, result);
}

int main()
{
   char str[200], target[50], replace[50];

   printf("Enter a string: ");
   scanf("%[^\n]",str);
   getchar();
   printf("Enter target substring: ");
   scanf("%[^\n]",target);
   getchar();
```

```
    printf("Enter replacement substring: ");
    scanf("%[^\n]",replace);
    getchar();

    findandreplace(str, target, replace);
    printf("Result: %s\n", str);
    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"
Answer:

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

void reverseWords(char str[]);


int main()
{
    char str[200];
    printf("Enter a sentence: ");
    scanf("%[^\n]",str);

    reverseWords(str);
    printf("Reversed Sentence: %s\n", str);
    return 0;
}

void reverseWords(char str[])
{
    char result[200] = "", *words[100];
    int count = 0;

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

```
    while (token)
    {
        words[count++] = token;
        token = strtok(NULL, " ");
    }

    for (int i = count - 1; i >= 0; i--)
    {
        strcat(result, words[i]);

        if (i > 0)
        strcat(result, " ");
    }
    strcpy(str, result);
}
```

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"
Answer:

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

char* longestrepeatingsubstring(char str[])
{
    int n = strlen(str), maxlen = 0;
    static char result[100];

    for (int len = 1; len < n; len++)
    {
        for (int i = 0; i <= n - len; i++)
        {
            char temp[100];
```

```
            strncpy(temp, &str[i], len);
            temp[len] = '\0';

            for (int j = i + 1; j <= n - len; j++)
            {
               if (strncmp(temp, &str[j], len) == 0 && len > maxlen)
               {
                  maxlen = len;
                  strcpy(result, temp);
               }
            }
         }
      }

      if(maxlen>0)
      {
      return result;
      }
      else
      {
      printf("\nNo repeating substring is present");
      }
}

int main()
{
      char str[200];
      printf("Enter a string: ");
      scanf("%[^\n]",str);
      getchar();
      printf("Longest Repeating Substring: %s\n", longestrepeatingsubstring(str));
      return 0;
}
```

DYNAMIC MEMORY ALLOCATION

```
#include<stdio.h>
#include<string.h>
#include<ctype.h>
#include<stdlib.h>
int main()
{
   int *ptr;
   int num,i;
      printf("Enter the number of elements ");
   scanf("%d",&num);
   printf("\n");
   printf("the number entered is n=%d\n",num);
    //Dynamically allocated memory for array
    ptr=(int*)malloc(num*sizeof(int));
     //check whether memory is allocated successfully
      if(ptr==NULL)
   {
      printf("Memory not allocated\n");
      exit(0);
   }
   else
   {
      printf("memory is allocated sucessfully\n");
   }

   //populating the array
   for(int i=0;i<num;i++)  {
      ptr[i]=i+1;   }
   //Displaying the array
   for(int i=0;i<num;i++){
      printf("%d ",ptr[i]);
}
   //free memory
   free(ptr);
   return 0;
}
```