***Assignment 9 || String In-built functions***

Arjun Patel – FRN13J1124/006

Q1) strlen

*#include* <stdio.h>

*#include* <string.h>

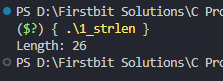
int main() {

    char str[] *=* "Hello, Firstbit solutions!";

    printf("Length: %d\n", strlen(str));

*return* 0;

}



Q2)strcpy

*#include* <stdio.h>

*#include* <string.h>

int main() {

    char src[] *=* "ArjunPatel";

    char dest[strlen(src)];

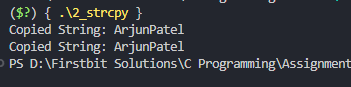
    strcpy(dest, src);

    printf("Copied String: %s\n", dest);

    printf("Copied String: %s\n", src);

*return* 0;

}



Q3) strncpy

*//copy only n char to dest str*

*#include* <stdio.h>

*#include* <string.h>

int main() {

    char src[] *=* "0123456789";

*// printf("%d", strlen(src));*

    char dest[strlen(src)];

    strncpy(dest, src, 5);

    dest[5] *=* '\0';

    printf("Copied String: %s\n", dest); *//01234*

*return* 0;

}



Q4) strcat

*#include* <stdio.h>

*#include* <string.h>

int main() {

    char str1[50] *=* "Arjun ";

    char str2[] *=* "Patel!";

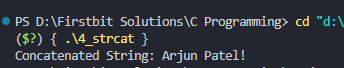
    strcat(str1, str2); *//it will concat str2 in str1*

    printf("Concatenated String: %s\n", str1); *//Arjun Patel!*

*// printf("Concatenated String: %s\n", str2); //Patel!*

*return* 0;

}



Q5) strncat

*#include* <stdio.h>

*#include* <string.h>

*//concat string upto n char*

int main() {

    char str1[50] *=* "Arjun ";

    char str2[] *=* "Patel";

    strncat(str1, str2, 3);

    printf("Concatenated String: %s\n", str1); *//Arjun Pat --> str1 + 3 char from str2*

*return* 0;

}



Q6) strcmp

*#include* <stdio.h>

*#include* <string.h>

*//return 1 if str1>str2, -1 is str1<str2, 0 if str1=str2*

int main() {

    char str1[] *=* "ccccc";

    char str2[] *=* "bbbbb";

    int result *=* strcmp(str1, str2);

*if* (result *<* 0)

        printf("str1 is less than str2\n"); *//this will executed*

*else* *if* (result *>* 0)

        printf("str1 is greater than str2\n");

*else*

        printf("str1 is equal to str2\n");

*return* 0;

}



Q8) strncmp

*#include* <stdio.h>

*#include* <string.h>

*//compare only n chars of str2*

int main() {

    char str1[] *=* "arjun";

    char str2[] *=* "arjunpatel";

    int n *=* 4;

    int result *=* strncmp(str1, str2, n);

*if* (result *==* 0)

        printf("The first %d characters are equal\n",n);

*else*

        printf("The first %d characters are not equal\n", n);

*return* 0;

}



Q9)strrchr 🡪 it will find last occurance of char in given string

*#include* <stdio.h>

*#include* <string.h>

*//it will find last occurance of char in given string*

int main() {

    char str[] *=* "Hello, World!";

    printf("base address %u\n", str); *//1994389066*

    char *\**ptr *=* strrchr(str, 'o');

    printf("%u\n", ptr); *//1994389084*

*if* (ptr)

        printf("Last occurrence at position: %ld\n", ptr *-* str); *//84-66 = 8*

*else*

        printf("Character not found\n");

*return* 0;

}



Q10)strstr -> Finds the first occurrence of a substring in a string.

*#include* <stdio.h>

*#include* <string.h>

*//Finds the first occurrence of a substring in a string.*

int main() {

    char str[] *=* "Arjun Patel";

    char find[] *=* "Patel";

    char *\**ptr *=* strstr(str, find);

*if* (ptr)

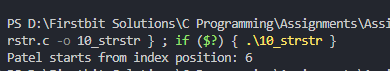
        printf("%s starts from index : %d\n",find,  ptr *-* str);

*else*

        printf("Substring not found\n");

*return* 0;

}



Q11) **strdup** -> Duplicates a string (non-standard, available in POSIX).

It will works with dynamic memory we can free memory after task completion.

*#include* <stdio.h>

*#include* <string.h>

*#include*<stdlib.h>

*//Duplicates a string (non-standard, available in POSIX).*

int main() {

    char str[] *=* "Hello";

    char *\**dup *=* strdup(str);

    char cpy[strlen(str)];

    strcpy(cpy, str);

    printf("Duplicated String: %s\n", cpy);

    printf("Duplicated String: %s\n", dup);

    free(dup); *// Remember to free memory*

    printf("Duplicated String: %s\n", dup);

*return* 0;

}



Q12) **strtok** -> Splits a string into tokens based on delimiters.

*#include* <stdio.h>

*#include* <string.h>

*// Splits a string into tokens based on delimiters.*

int main() {

    char str[] *=* "Hello World! C is more, fun.";

    printf("%u \n",str);

    char *\**token *=* strtok(str, " ,.!");

    printf("value of token %u \n", token);

*while* (token) {

        printf("%s\n", token);

        printf("%u\n", token);

        token *=* strtok(NULL, " #,.!"); *// Get next token*

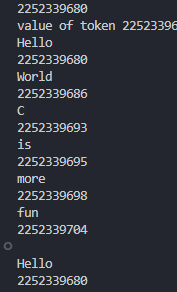
    }

    printf("\n%s\n", str);

    printf("%u\n", str);

*return* 0;

}



Q13) memstr

*#include* <stdio.h>

*#include* <string.h>

*//Fills a string (or memory block) with a specific character.*

int main() {

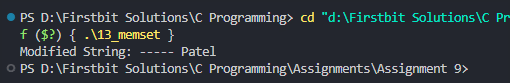
    char str[20] *=* "Arjun Patel";

    memset(str, '-', 5); *// Replace first 3 characters with '-'*

    printf("Modified String: %s\n", str);

*return* 0;

}



Q14) memcap

*#include* <stdio.h>

*#include* <string.h>

int main() {

    char str1[] *=* "Hello";

    char str2[] *=* "Hello";

    int result *=* memcmp(str1, str2, 5);

*if* (result *==* 0)

        printf("Blocks are identical\n");

*else*

        printf("Blocks are different\n");

*return* 0;

}



Q15) toupper (char)

*#include*<stdio.h>

*#include*<ctype.h>

*#include*<string.h>

int main(){

    char str[] *=* "Arjun Patel";

    int n *=* strlen(str);

*// printf("%d \n", n);*

    printf("%d \n", strlen(str));

    char str2[strlen(str)*+*1];

*// printf("%d \n", strlen(str2));*

*// printf("%c", toupper(c));*

*for* (int i *=* 0; i *<* strlen(str); i*++*)

    {

        str2[i] *=* toupper(str[i]);

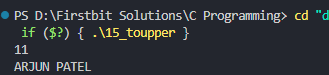
    }

    str2[strlen(str)] *=* '\0';

    printf("%s", str2);

*return* 0;

}



Q16) tolower(char)

*#include*<stdio.h>

*#include*<ctype.h>

*#include*<string.h>

int main(){

    char str[] *=* "ARJUN Patel";

    int n *=* strlen(str);

*// printf("%d \n", n);*

    printf("%d \n", strlen(str));

    char str2[strlen(str)*+*1];

*// printf("%d \n", strlen(str2));*

*// printf("%c", toupper(c));*

*for* (int i *=* 0; i *<* strlen(str); i*++*)

    {

        str2[i] *=* tolower(str[i]);

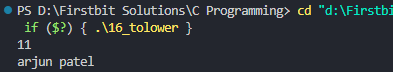
    }

    str2[strlen(str)] *=* '\0';

    printf("%s", str2);

*return* 0;

}



Q17) strupr

*#include*<stdio.h>

*#include*<string.h>

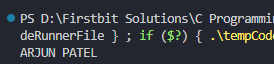
int main(){

    char str[] *=* "Arjun Patel";

    printf("%s", strupr(str));

*return* 0;

}



Q18) strlwr

*#include*<stdio.h>

*#include*<string.h>

int main(){

    char str[] *=* "ARJUN PATEL";

    printf("%s", strlwr(str));

*return* 0;

}



Q19) strpbrk :

// strpbrk(const char \*str, const char \*accept)

// Finds first character matching any in accept

*#include*<stdio.h>

*#include*<string.h>

*// strpbrk(const char \*str, const char \*accept)*

*// Finds first character matching any in accept*

int main(){

    char str[] *=* "Arjun Patel";

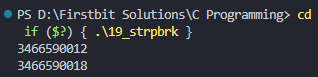
    char *\**pos *=* strpbrk(str, "Patel"); *// points to 'p'*

    printf("%u \n", str);

    printf("%u \n", pos);

*return* 0;

}



Q20) strspn

// Gets length of initial segment matching accept

// cCopysize\_t len = strspn("Hello", "Hel"); // returns 3

*// strspn(const char \*str, const char \*accept)*

*// Gets length of initial segment matching accept*

*// cCopysize\_t len = strspn("Hello", "Hel");  // returns 3*

*#include*<stdio.h>

*#include*<string.h>

 int main(){

    char str[] *=* "Arjun Patel";

    int len *=* strspn(str, "Arjun b");

    printf("%d", len);

*return* 0;

 }



Q21) strcspn

*// strcspn(const char \*str, const char \*reject)*

*// Gets length of initial segment not matching reject*

*// cCopysize\_t len = strcspn("Hello", "o");  // returns 4*

*#include*<stdio.h>

*#include*<string.h>

int main(){

    char str[] *=* "Arjun Patel";

    int len *=* strcspn(str, "P");

    printf("%d", len);

*return* 0;

}



Q22) memmove

*// memmove(void \*dest, const void \*src, size\_t n)*

*// Copies n bytes (safe for overlapping regions)*

*// cCopychar str[] = "Hello";*

*// memmove(str+1, str, 4);*

*//A r j u n   P a t e l*

*//0 1 2 3 4 5 6 7 8 9 10*

*#include*<stdio.h>

*#include*<string.h>

int main(){

    char str[] *=* "Arjun Patel";

    memmove(str*+*1, str*+*6, 5);

*// memmove(dest, src, 6);*

*//its replaces*

    printf("%s", str);

*return* 0;

}



Q23) memcmp

*// memcmp(const void \*str1, const void \*str2, size\_t n)*

*// Compares n bytes of memory*

*// cCopychar s1[] = "Hello";*

*// char s2[] = "Help";*

*// int result = memcmp(s1, s2, 4);*

*#include*<stdio.h>

*#include*<string.h>

int main(){

    char str[] *=* "Arjun Patel";

    char str2[] *=* "Arjsf";

    int res *=* memcmp(str, str2, 4);

*//memcmp returns 0 if 4 bytes is same as cmp string, else returns 1*

    printf("%d \n", res);

*return* 0;

}



Q24) memcpy

*// memcpy(void \*dest, const void \*src, size\_t n)*

*// Copies n bytes from src to dest*

*// cCopychar src[] = "Hello";*

*// char dest[6];*

*// memcpy(dest, src, 6);*

*#include*<stdio.h>

*#include*<string.h>

int main(){

    char str[] *=* "Hello world";

    char dest[strlen(str)*+*1];

    memcpy(dest, str, 4);

    dest[4] *=* '\0';

    printf("%s \n", dest);

*return* 0;

}



Q25)sprint

*// sprintf(char \*str, const char \*format, ...)*

*// Writes formatted data to string*

*// cCopychar str[50];*

*// sprintf(str, "Value is %d", 123);*

*#include*<stdio.h>

*#include*<string.h>

int main(){

    char str[] *=* "Arjun Patel";

    sprintf(str, "value is %d", 123);

    printf("%s", str);

*return* 0;

}



Q26) snprintf

*// snprintf(char \*str, size\_t size, const char \*format, ...)*

*// Writes formatted data to string with size limit*

*// cCopychar str[50];*

*// snprintf(str, 50, "Value is %d", 123);*

*#include*<stdio.h>

*#include*<string.h>

int main(){

    char str[] *=* "Arjun Patel";

    snprintf(str, 12, "Value is %d", 12345);

    printf("%s", str);

*return* 0;

}



***------END------***