

Output:

Enter 2 strings:

person
child

length of string: 6

Concatenated: personchild

Comparison of strings: 13

Enter substring to insert: kid

Enter position to insert substring at: 2

After substring insertion: pekidlsor

Enter substring to delete: ss

Substring found at index 2!

After substring removal: peon

Lab Sheet 3

1. Write a program to perform the following operations without using string handling functions:
- length of the string
 - string concatenation
 - string comparison
 - to insert a sub string
 - to delete a substring

Source Code:

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

```
int str_len(char* str) {
    int i;
    for(i=0; str[i]; i++);
    return i;
}
```

```
int str_cmp(char* str, char* str1) {
    int i=0;
    while(str[i] && str1[i]) {
        if(str[i] != str1[i])
            return str[i] - str1[i];
        i++;
    }
}
```

Teacher's Signature _____

E bilden

gewollte und erwartete
verbale Worte. Sollten sie nicht
wiederholen, so kann man
dieses wiederholen.
Wiederholung kann (1)
verbale Worte wiederholen.
Sollten sie wiederholen (2)
so kann man wiederholen.

Abbildung

<Abbildung> Abbildung

(die *reis) und die Reise
ist hier
(+reisdr. (O=)) von
in unserer

(die *reis die *reise und die Reise)

(Füller ist [Füller] wieder
(Füller = Füller ist
Füller - Füller wieder ist)

```
return str[i] - str1[i];
```

}

```
int main() {
    int i, j, pos, substr_pos;
    char str[100], str1[100], substr[100];

    printf("Enter 2 strings: \n");
    scanf("%s", str);
    scanf("%s", str1);

    printf("length of str: %d\n", str_len(str));

    char str2[str_len(str) + str_len(str1) + 1];
    for (i = 0; str[i]; i++) str2[i] = str[i];
    for (i = 0; str1[i]; i++) str2[str_len(str) + i] = str1[i];
    str2[str_len(str) + str_len(str1)] = '\0';

    printf("Concatenated: %s\n", str2);
    printf("Comparison of strings: %d\n", str_cmp(str, str1));

    printf("Enter substring to insert: ");
    scanf("%s", substr);
    printf("Enter position to insert substring at: ");
    scanf("%d", &pos);

    char str3[str_len(str) + str_len(substr) + 1];
}
```

Wahr - Falsch

Wahr - Falsch
Wahr - Falsch
Wahr - Falsch

(L1) wahr & falsch) Wahr
(L2 "Falsch") Falsch
(L3 "Wahr") Wahr

Wahr & der "Falsch" ist falsch)

$L + (L1) \text{ wahr} + (L2) \text{ falsch} \vdash L$ als
 $L \text{ ist } L1 \text{ oder } L2 \text{ ist } L$
 $L \text{ ist } L1 \text{ und } L2 \text{ ist } L$ (Falsch ist falsch) ist
 $L = [(L1) \text{ wahr} + (L2) \text{ falsch}] \text{ ist}$

(L2 "Falsch" ist falsch) Wahr
Falsch "ist falsch" ist wahr
"L ist L"

"L ist wahr & falsch ist falsch" Wahr
Falsch ist falsch & wahr ist falsch Wahr

(wahr "betr") Falsch

$L + (L1 \text{ wahr} + L2) \text{ falsch} \vdash L$ als
Falsch ist falsch

```

for(i=0; i< pos; i++) str3[i] = str[i];
for(i=0; substr[i]; i++) str3[pos+i] = substr[i];
for(i=pos; str[i]; i++) str3[str_len(substr)+i] =
str[i];
str3[str_len(str) + str_len(substr)] = '\0';

```

printf ("After substring insertion: %s\n", str3);

printf ("Enter substring to delete: ");
scanf ("%s", substr);

substr-pos = -1;

```

for(i=0; str[i]; i++) {
    int matching = 1;

```

```

        for(j=0; substr[j]; j++)
            if(substr[j] != str[i+j]) matching = 0;

```

```

        if(matching == 1) {

```

substr-pos = i;

break;

}

```

if(substr-pos == -1) {

```

```

    printf ("Substring not found in string\n");
    return 1;

```

}


```
printf("Substring found at index %d!\n", substr_pos);
```

```
for(i=0; substr[i]; i++)
```

```
    for(j=substr_pos; str[j]; j++)
```

```
        str[j] = str[j+1];
```

```
printf("After substring removal: %s\n", str);
```

```
return 0;
```

```
}
```

Output:

Enter number of students in school: 3
Enter roll number of student: 3
Enter name of student: Jerome
Enter grade of student: A
Enter roll number of student: 2
Enter name of student: Kalp
Enter grade of student: A
Enter roll number of student: 1
Enter name of student: Parav
Enter grade of student: A

Name	Roll no.	Grade
Parav	1	A
Kalp	2	A
Jerome	3	A

2. Write a C program to define a student structure with the data members to store name, roll no. & grade of the student. Also write the required functions to read, display, and sort student information according to the roll number of the student. All the member functions will have an array of objects as arguments.

Source Code:

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

```
typedef struct {
    char name[50];
    int roll-no;
    char grade;
} Student;
```

```
void read (Student * students, int n) {
    int i;
    for (i = 0; i < n; i++) {
        printf ("Enter roll number of student: ");
        scanf ("%d", & students[i].roll-no);
        printf ("Enter name of student: ");
        scanf ("%s", students[i].name);
        getchar();
        printf ("Enter grade of student: ");
        scanf ("%c", & students[i].grade);
    }
}
```

Teacher's Signature _____

and the students are very
good. Some students have
done well, others have
done poorly. There is
a lot of room for improvement.
The students are
very good.

Students' Attitudes

The students' attitudes
towards learning
language are
mixed. Some
are very
interested
in learning
language.

The teacher, students & teacher have been

Attitudes of Students

The students' attitudes towards the teacher
are mixed. Some like the teacher
and some do not like her.

The students' attitudes towards the language
are mixed. Some like it and some do not like it.

The students' attitudes towards the class
are mixed. Some like it and some do not like it.

The students' attitudes towards the school
are mixed. Some like it and some do not like it.

The students' attitudes towards the teacher
are mixed. Some like her and some do not like her.

```

    }
}

void sort (Student * students, int n) {
    int i, j;
    for (i = 0; i < n - 1; i++) {
        for (j = 0; j < n - i - 1; j++) {
            if (students[j].roll_no > students[j + 1].roll_no) {
}

```

~~Student~~ temp = students[j];
 students[j] = students[j + 1];
 students[j + 1] = temp;
}

{

{

{

```

void display (Student * students, int n) {
    int i;
    printf ("Name \t \t Roll no. \t \t Grade \n");
    for (i = 0; i < n; i++) {
        printf ("%s \t \t %d \t \t %c \n", students[i].name,
               students[i].roll_no, students[i].grade);
}

```

int main () {

int n;

printf ("Enter number of students in school: ");
scanf ("%d", &n);

1. (1) $\frac{1}{2} \times 10^6$ (1000000) $\times 10^3$ m^3
= 500000000 m^3

2. $\frac{1}{2} \times 10^6$ (1000000) $\times 10^3$ m^3
 $\times 10^3$ kg/m^3
= 500000000 kg

3. $\frac{1}{2} \times 10^6$ (1000000) $\times 10^3$ m^3
 $\times 10^3$ kg/m^3
 $\times 10^3$ J/kg
= 500000000 J

4. $\frac{1}{2} \times 10^6$ (1000000) $\times 10^3$ m^3
 $\times 10^3$ kg/m^3
 $\times 10^3$ J/kg
 $\times 10^3$ N
= 500000000 N

5. $\frac{1}{2} \times 10^6$ (1000000) $\times 10^3$ m^3
 $\times 10^3$ kg/m^3
 $\times 10^3$ J/kg
 $\times 10^3$ N
 $\times 10^3$ s
= 500000000 W

6. $\frac{1}{2} \times 10^6$ (1000000) $\times 10^3$ m^3
 $\times 10^3$ kg/m^3
 $\times 10^3$ J/kg
 $\times 10^3$ N
 $\times 10^3$ s
 $\times 10^3$ A
= 500000000 V

7. $\frac{1}{2} \times 10^6$ (1000000) $\times 10^3$ m^3
 $\times 10^3$ kg/m^3
 $\times 10^3$ J/kg
 $\times 10^3$ N
 $\times 10^3$ s
 $\times 10^3$ V
= 500000000 F

8. $\frac{1}{2} \times 10^6$ (1000000) $\times 10^3$ m^3
 $\times 10^3$ kg/m^3
 $\times 10^3$ J/kg
 $\times 10^3$ N
 $\times 10^3$ s
 $\times 10^3$ V
 $\times 10^3$ A
= 500000000 Wb

Student students[n];

read(students, n);

sort(students, n);

display(students, n);

return 0;

3

Teacher's Signature _____

Output:

Enter number of students in school: 3
Enter roll number of student: 3
Enter name of student: Rahul
Enter marks of student: 50
Enter roll number of student: 2
Enter name of student: Vaibhav
Enter marks of student: 70
Enter roll number of student: 1
Enter name of student: XYZ
Enter marks of student: 60

Name	Roll no.	Marks
Rahul	3	50.000000
Vaibhav	2	70.000000
XYZ	1	60.000000

Details of students with highest marks:

Name	Roll no.	Marks
Vaibhav	2	70.000000

3. Define a structure Student with the following members:
char name [50] - to store student name as string
int roll-no - to store roll no.
float marks - to store marks

Write a C program that:

- (i) Reads the details of 'n' students using a function that uses pointer to structure as an argument.
- (ii) Displays the details of all students using a separate function.
- (iii) Finds and displays the student with the highest marks using pointer-based access.

Source Code:

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

```
typedef struct {  
    char name[50];  
    int roll-no;  
    float marks;  
} Student;
```

```
void read (Student * students, int n) {  
    int i;  
    for (i = 0; i < n; i++) {  
        printf ("Enter roll number of student: ");
```

Teacher's Signature _____


```

scanf("%d", &(students + i) → roll_no);
printf("Enter name of student: ");
scanf("%s", &(students + i) → name);
getchar();
printf("Enter marks of student: ");
scanf("%f", &(students + i) → marks);
}
}

```

```

void display(Student * students, int n) {
    int i;
    printf("Name \t\t Roll no. \t\t Marks \n");
    for(i = 0; i < n; i++)
        printf("%s \t\t %d \t\t %f \n", (students + i)
    → name, (students + i) → roll_no, (students + i) →
    marks);
}

```

```

void highest(Student * students, int n) {
    Student high;
    high.marks = -1;
    int i;
    for(i = 0; i < n; i++) {
        if((students + i) → marks > high.marks) {
            high = *(students + i);
        }
    }
}

```

```

Student high Arr[] = {high};
printf("Details of students with highest marks: \n");

```

Teacher's Signature _____

} display (highArr, 1);

```
int main() {
    int n;
    printf("Enter number of students in school: ");
    scanf("%d", &n);
    Student students[n];
    read(students, n);
    display (students, n);
    highest (students, n);
    return 0;
}
```