

## C SOURCE CODE:

```
#include <stdio.h>
void sort(int *a, int n) {
    for (int i = 0; i < n - 1; i++) {
        int min_ind = i;
        for (int j = i + 1; j < n; j++) {
            if (a[j] < a[min_ind]) min_ind = j;
        }
        int temp = a[i];
        a[i] = a[min_ind];
        a[min_ind] = temp;
    }
}
void rearrange(int *a, int n) {
    sort(a, n);
    int temp[n];
    int p = 0, q = n - 1;
    for (int i = 0; i < n; i++) {
        if (i % 2 == 0)
            temp[i] = a[p++];
        else
            temp[i] = a[q--];
    }
    for (int i = 0; i < n; i++) a[i] = temp[i];
}

int main() {
    int n;
    printf("Enter size of array: ");
    scanf("%d", &n);
    int a[n];
    printf("Enter the elements of the array:\n");
    for (int i = 0; i < n; i++) scanf("%d", &a[i]);

    rearrange(a, n);

    printf("Rearranged array is:\n");
    for (int i = 0; i < n; i++) printf("%d ", a[i]);
}
```

## OUTPUT:

```
Enter size of array: 6
Enter the elements of the array:
12
6
3
7
3
2
Rearranged array is:
2 12 3 7 3 6
Process returned 0 (0x0)    execution time : 4.953 s
Press any key to continue.
_
```

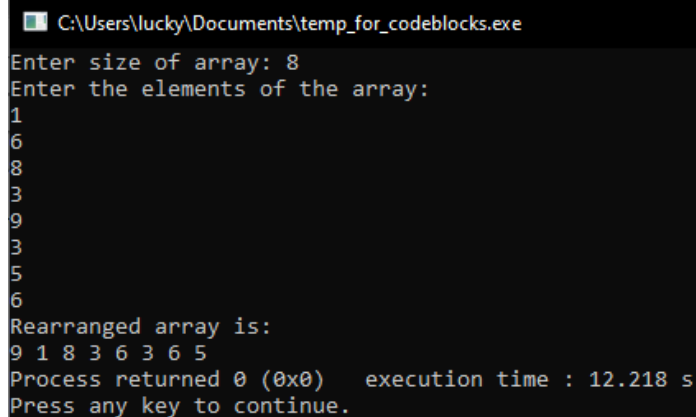
## C SOURCE CODE:

```
#include <stdio.h>
void sort(int *a, int n) {
    for (int i = 0; i < n - 1; i++) {
        int min_ind = i;
        for (int j = i + 1; j < n; j++) {
            if (a[j] < a[min_ind]) {
                min_ind = j;
            }
        }
        int temp = a[i];
        a[i] = a[min_ind];
        a[min_ind] = temp;
    }
}
void rearrange(int *a, int n) {
    sort(a, n);
    int temp[n];
    int p = 0, q = n - 1;

    for (int i = 0; i < n; i++) {
        if (i % 2 == 0)
            temp[i] = a[q--];
        else
            temp[i] = a[p++];
    }

    for (int i = 0; i < n; i++) a[i] = temp[i];
}
int main() {
    int n;
    printf("Enter size of array: ");
    scanf("%d", &n);
    int a[n];
    printf("Enter the elements of the array:\n");
    for (int i = 0; i < n; i++) scanf("%d", &a[i]);
    rearrange(a, n);
    printf("Rearranged array is:\n");
    for (int i = 0; i < n; i++) printf("%d ", a[i]);
}
```

## OUTPUT:



```
C:\Users\lucky\Documents\temp_for_codeblocks.exe
Enter size of array: 8
Enter the elements of the array:
1
6
8
3
9
3
5
6
Rearranged array is:
9 1 8 3 6 3 6 5
Process returned 0 (0x0)   execution time : 12.218 s
Press any key to continue.
```

## C SOURCE CODE:

```
#include <stdio.h>
#include <stdlib.h>
int main() {
    int row, col;
    printf("Enter the number of rows: ");
    scanf("%d", &row);

    printf("Enter the number of columns: ");
    scanf("%d", &col);

    int matrix[row][col];
    printf("Enter the elements of the matrix:\n");
    for (int i = 0; i < row; i++) {
        for (int j = 0; j < col; j++) scanf("%d", &matrix[i][j]);
    }
    printf("\nOriginal matrix:\n");
    for (int i = 0; i < row; i++) {
        for (int j = 0; j < col; j++) printf("%d ", matrix[i][j]);
        printf("\n");
    }
    printf("\nMirror matrix:\n");
    for (int i = 0; i < row; i++) {
        for (int j = col - 1; j >= 0; j--) printf("%d ", matrix[i][j]);
        printf("\n");
    }
    printf("\nTranspose matrix:\n");
    for (int i = 0; i < col; i++) {
        for (int j = 0; j < row; j++) printf("%d ", matrix[j][i]);
        printf("\n");
    }
}
```

## OUTPUT:

```
Enter the number of rows: 3
Enter the number of columns: 4
Enter the elements of the matrix:
1
2
3
4
5
6
7
8
9
0
11
12

Original matrix:
1 2 3 4
5 6 7 8
9 0 11 12

Mirror matrix:
4 3 2 1
8 7 6 5
12 11 0 9

Transpose matrix:
1 5 9
2 6 0
3 7 11
4 8 12

Process returned 0 (0x0)   execution time : 13.073 s
Press any key to continue.
```

## C SOURCE CODE:

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

int main() {
    int row1, col1, row2, col2;
    printf("Enter the number of rows in matrix 1: ");
    scanf("%d", &row1);
    printf("Enter the number of columns in matrix 1: ");
    scanf("%d", &col1);
    printf("Enter the number of rows in matrix 2: ");
    scanf("%d", &row2);
    printf("Enter the number of columns in matrix 2: ");
    scanf("%d", &col2);
    if (col1 != row2) {
        printf(
            "Matrix multiplication is not possible (Column of matrix1 should equal "
            "Row of matrix2).\n");
        return 0;
    }
    int matrix1[row1][col1], matrix2[row2][col2], result[row1][col2];
    printf("Enter the elements of matrix 1:\n");
    for (int i = 0; i < row1; i++) {
        for (int j = 0; j < col1; j++) scanf("%d", &matrix1[i][j]);
    }
    printf("Enter the elements of matrix 2:\n");
    for (int i = 0; i < row2; i++) {
        for (int j = 0; j < col2; j++) scanf("%d", &matrix2[i][j]);
    }
    for (int i = 0; i < row1; i++) {
        for (int j = 0; j < col2; j++) {
            result[i][j] = 0;
            for (int k = 0; k < col1; k++)
                result[i][j] += matrix1[i][k] * matrix2[k][j];
        }
    }
    printf("\nOriginal matrix 1:\n");
    for (int i = 0; i < row1; i++) {
        for (int j = 0; j < col1; j++) printf("%d ", matrix1[i][j]);
        printf("\n");
    }
    printf("\nOriginal matrix 2:\n");
    for (int i = 0; i < row2; i++) {
        for (int j = 0; j < col2; j++) printf("%d ", matrix2[i][j]);
        printf("\n");
    }
    printf("\nResultant matrix:\n");
    for (int i = 0; i < row1; i++) {
        for (int j = 0; j < col2; j++) printf("%d ", result[i][j]);
        printf("\n");
    }
}
```

OUTPUT:

```
Enter the number of rows in matrix 1: 3
Enter the number of columns in matrix 1: 4
Enter the number of rows in matrix 2: 4
Enter the number of columns in matrix 2: 2
Enter the elements of matrix 1:
```

```
1
2
3
45
5
6
7
8
9
10
11
12
```

```
Enter the elements of matrix 2:
```

```
1
2
3
4
5
6
7
8
```

```
Original matrix 1:
```

```
1 2 3 45
5 6 7 8
9 10 11 12
```

```
Original matrix 2:
```

```
1 2
3 4
5 6
7 8
```

```
Resultant matrix:
```

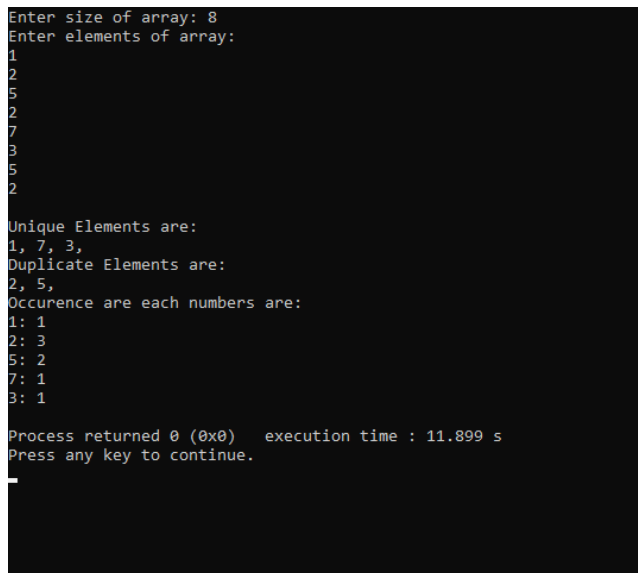
```
337 388
114 140
178 220
```

## C SOURCE CODE:

```
#include <stdio.h>
void countFreq(int arr[], int n) {
    int count[n];
    for (int i = 0; i < n; i++) count[i] = -1;
    for (int i = 0; i < n - 1; i++) {
        if (count[i] == 0) continue;
        int c = 1;
        for (int j = i + 1; j < n; j++) {
            if (arr[i] == arr[j]) {
                count[j] = 0;
                c++;
            }
        }
        count[i] = c;
    }
    printf("\nUnique Elements are:\n");
    for (int i = 0; i < n; i++) {
        if (count[i] == 1) printf("%d, ", arr[i]);
    }
    printf("\nDuplicate Elements are:\n");
    for (int i = 0; i < n; i++) {
        if (count[i] > 1) printf("%d, ", arr[i]);
    }
    printf("\nOccurrence are each numbers are:\n");
    for (int i = 0; i < n; i++) {
        if (count[i] != 0) printf("%d: %d\n", arr[i], count[i]);
    }
}

int main() {
    int n;
    printf("Enter size of array: ");
    scanf("%d", &n);
    int arr[n];
    printf("Enter elements of array: \n");
    for (int i = 0; i < n; i++) scanf("%d", &arr[i]);
    countFreq(arr, n);
}
```

## OUTPUT:



```
Enter size of array: 8
Enter elements of array:
1
2
5
2
7
3
5
2

Unique Elements are:
1, 7, 3,
Duplicate Elements are:
2, 5,
Occurrence are each numbers are:
1: 1
2: 3
5: 2
7: 1
3: 1

Process returned 0 (0x0)   execution time : 11.899 s
Press any key to continue.
_
```

## C SOURCE CODE:

```
#include <stdio.h>
void insertionSort(int a[], int N) {
    for (int i = 1; i < N; i++) {
        int key = a[i];
        int j = i - 1;
        while (j >= 0 && a[j] > key) {
            a[j + 1] = a[j];
            j--;
        }
        a[j + 1] = key;
    }
}

int main() {
    int N;
    printf("Enter the size of array: ");
    scanf("%d", &N);

    int a[N];
    printf("Enter %d numbers: ", N);
    for (int i = 0; i < N; i++) {
        scanf("%d", &a[i]);
    }

    insertionSort(a, N);

    printf("\nSorted numbers: ");
    for (int i = 0; i < N; i++) {
        printf("%d ", a[i]);
    }
    printf("\n");
}
```

## OUTPUT:

```
Enter the size of array: 7
Enter 7 numbers: 2
6
4
2
1
8
4

Sorted numbers: 1 2 2 4 4 6 8

Process returned 0 (0x0)   execution time : 5.866 s
Press any key to continue.
```

## C SOURCE CODE:

```
#include <stdio.h>

int binary_search(int a[], int n, int x) {
    int low = 0;
    int high = n - 1;
    while (low <= high) {
        int mid = low + (high - low) / 2;
        if (a[mid] == x) {
            return mid;
        } else if (a[mid] < x) {
            low = mid + 1;
        } else {
            high = mid - 1;
        }
    }
    return -1;
}

int main() {
    int n;
    printf("Enter size of array: ");
    scanf("%d", &n);
    int arr[n];
    printf("Enter elements of array: \n");
    for (int i = 0; i < n; i++) scanf("%d", &arr[i]);
    int x;
    printf("Enter a number to search for: ");
    scanf("%d", &x);

    int index = binary_search(arr, n, x);
    if (index == -1) {
        printf("Number not found\n");
    } else {
        printf("Number found at index %d\n", index);
    }
}
```

## OUTPUT:

```
Enter size of array: 9
Enter elements of array:
1
3
5
6
7
8
9
10
114
Enter a number to search for: 8
Number found at index 5

Process returned 0 (0x0)   execution time : 12.288 s
Press any key to continue.
```



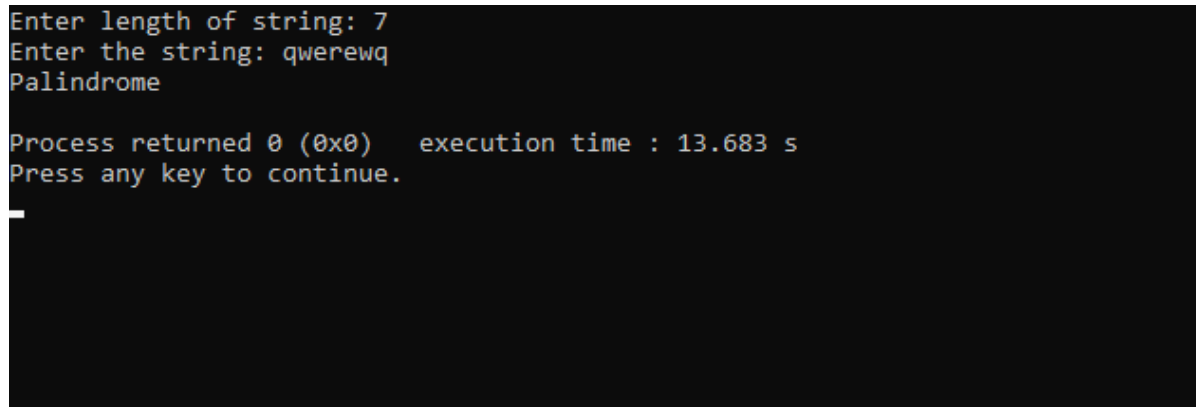
## C SOURCE CODE:

```
#include <stdio.h>

int main() {
    int n;
    printf("Enter length of string: ");
    scanf("%d", &n);
    char s[n];
    printf("Enter the string: ");
    scanf("%s", s);

    int i = 0;
    int j = n - 1;
    while (i <= j) {
        if (s[i] != s[j]) {
            printf("Not a palindrome\n");
            return 0;
        }
        i++;
        j--;
    }
    printf("Palindrome\n");
}
```

## OUTPUT:



```
Enter length of string: 7
Enter the string: qwerewq
Palindrome

Process returned 0 (0x0)   execution time : 13.683 s
Press any key to continue.
_
```

### PROBLEM STATEMENT 9:

Write a program in C to search a substring within a string and replace it with another string.

### C SOURCE CODE:

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

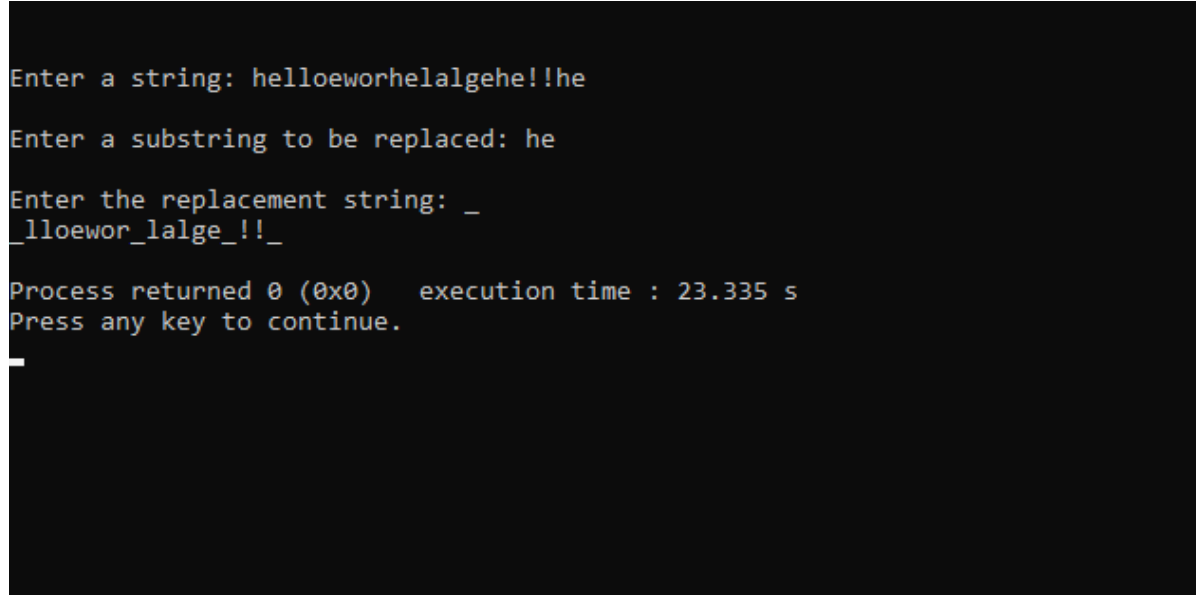
int main() {
    char str[100], sub[100], new[100] = "", replace[100];
    printf("\n\nEnter a string: ");
    gets(str);

    printf("\nEnter a substring to be replaced: ");
    gets(sub);

    printf("\nEnter the replacement string: ");
    gets(replace);

    int len = strlen(sub);
    for (int i = 0; i < strlen(str); i++) {
        if (strncmp(&str[i], sub, len) == 0) {
            strcat(new, replace);
            i += len - 1;
        } else
            strncat(new, &str[i], 1);
    }
    puts(new);
}
```

### OUTPUT:



```
Enter a string: helloeworhelalgehe!!he
Enter a substring to be replaced: he
Enter the replacement string: _
_lloewor_lalge_!!_
Process returned 0 (0x0)   execution time : 23.335 s
Press any key to continue.
_
```

#### PROBLEM STATEMENT 10:

WAP to create a structure Student with (name, subject, roll, sid, marks) using appropriate data types, then take input and display the values of member variable of structure variable.

#### C SOURCE CODE:

```
#include <stdio.h>
struct student {
    char name[50];
    char subject[50];
    int roll;
    int sid;
    float marks;
};
int main() {
    struct student s;
    printf("Enter name: ");
    scanf("%s", s.name);
    printf("Enter subject: ");
    scanf("%s", s.subject);
    printf("Enter roll: ");
    scanf("%d", &s.roll);
    printf("Enter SID: ");
    scanf("%d", &s.sid);
    printf("Enter marks: ");
    scanf("%f", &s.marks);
    printf("\nName: %s\nSubject: %s\nRoll: %d\nSID: %d\nMarks:%f\n", s.name,
        s.subject, s.roll, s.sid, s.marks);
}
```

#### OUTPUT:

```
Enter name: test
Enter subject: phy
Enter roll: 26
Enter SID: 1
Enter marks: 95

Name: test
Subject: phy
Roll: 26
SID: 1
Marks:95.000000

Process returned 0 (0x0)   execution time : 6.482 s
Press any key to continue.
```

## PROBLEM STATEMENT 11:

Write a Menu driven program to perform the following operations on array of structure of above (Student) data type:

1. Insert
2. Display
3. Delete
4. Search

## C SOURCE CODE:

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#define n 10

struct student {
    char name[20];
    char subject[20];
    int roll;
    int sid;
    int marks;
} typedef student;

student s[n];
int filled[n];

void insert() {
    int i = 0;
    while (filled[i] == 1 && i < n) i++;
    if (i == n) {
        printf("No position left!!! Delete an entry!!!");
        return;
    }

    printf("\nEnter the name of student %d: ", i + 1);
    scanf("%s", s[i].name);
    printf("\nEnter the subject of student %d: ", i + 1);
    scanf("%s", s[i].subject);
    printf("\nEnter the roll no. of student %d: ", i + 1);
    scanf("%d", &s[i].roll);
    printf("\nEnter the sid of student %d: ", i + 1);
    scanf("%d", &s[i].sid);
    printf("\nEnter the marks of student %d: ", i + 1);
    scanf("%d", &s[i].marks);
    filled[i] = 1;
}

void display() {
    int i;
    for (i = 0; i < n; i++) {
        if (filled[i] == 0) continue;
        printf("\nName of student %d: %s", i + 1, s[i].name);
        printf("\nSubject of student %d: %s", i + 1, s[i].subject);
        printf("\nRoll no. of student %d: %d", i + 1, s[i].roll);
        printf("\nSid of student %d: %d", i + 1, s[i].sid);
        printf("\nMarks of student %d: %d", i + 1, s[i].marks);
    }
}
```

```

    }
}

void delete() {
    int i, j, k;
    int id;
    int deleted = 0;
    printf("\nEnter the id of student to be deleted: ");
    scanf("%d", &id);
    for (i = 0; i < n; i++) {
        if (filled[i] == 0) continue;
        if (id == s[i].sid) {
            filled[i] = 0;
            deleted = 1;
        }
    }
    if (!deleted) {
        printf("\nStudent not found");
    } else {
        printf("\nStudent deleted");
    }
}

void search() {
    int i, j, k;
    int id;
    printf("\nEnter the name of ID to be searched: ");
    scanf("%d", &id);
    for (i = 0; i < n; i++) {
        if (filled[i] == 0) continue;
        if (s[i].sid == id) {
            printf("\nName of student %d: %s", i + 1, s[i].name);
            printf("\nSubject of student %d: %s", i + 1, s[i].subject);
            printf("\nRoll no. of student %d: %d", i + 1, s[i].roll);
            printf("\nSid of student %d: %d", i + 1, s[i].sid);
            printf("\nMarks of student %d: %d", i + 1, s[i].marks);
            break;
        }
    }
    if (i == n) {
        printf("\nStudent not found");
    }
}

int main() {
    int i, j, k, ch;
    for (int i = 0; i < n; i++) filled[i] = 0;

    while (1) {
        printf(
            "\n1. Insert\n2. Display\n3. Delete\n4. Search\n5. Exit\nEnter your "
            "choice: ");
        scanf("%d", &ch);
        switch (ch) {
            case 1:
                insert();
                break;
            case 2:
                display();

```

```

        break;
    case 3:
        delete ();
        break;
    case 4:
        search();
        break;
    case 5:
        return 0;
    default:
        printf("\nInvalid choice");
    }
}
}

```

OUTPUT:

```

1. Insert
2. Display
3. Delete
4. Search
5. Exit
Enter your choice: 1

Enter the name of student 1: ayush
Enter the subject of student 1: phy
Enter the roll no. of student 1: 2
Enter the sid of student 1: 1
Enter the marks of student 1: 96

1. Insert
2. Display
3. Delete
4. Search
5. Exit
Enter your choice: 2

Name of student 1: ayush
Subject of student 1: phy
Roll no. of student 1: 2
Sid of student 1: 1
Marks of student 1: 96

1. Insert
2. Display
3. Delete
4. Search
5. Exit
Enter your choice: 5

Process returned 0 (0x0)   execution time : 26.877 s
Press any key to continue.

```

## PROBLEM STATEMENT 12:

WAP to implement Single linked list using the following menu driven functions.

1. Insert at Beginning
2. Insert at End
3. Insert at specific position
4. Display
5. Delete
6. Reverse Display
7. Reverse the linked list
8. Search
9. Sort (using selection sort)

## C SOURCE CODE:

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

struct node {
    int data;
    struct node* next;
};

struct node* HEAD = NULL;
int size = 0;

void insertBeginning() {
    int data;
    printf("Enter the data to enter: ");
    scanf("%d", &data);
    insertPos(1, data);
}

void insertEnd() {
    int data;
    printf("Enter the data to enter: ");
    scanf("%d", &data);
    insertPos(size + 1, data);
}

void insertPos(int pos, int data) {
    if (pos < 1 || pos > size + 1) {
        printf("INVALID POSITION\n");
        return;
    }
    struct node* link = HEAD;
    struct node* toAdd = malloc(sizeof(struct node));
    toAdd->data = data;
    toAdd->next = NULL;

    if (pos == 1) {
        toAdd->next = HEAD;
        HEAD = toAdd;
        size++;
        return;
    }
    int current_pos = 1;
    while (1) {
```

```

        if (current_pos == pos - 1) {
            toAdd->next = link->next;
            link->next = toAdd;
            size++;
            return;
        }
        link = link->next;
        current_pos++;
    }
}

void printList() {
    struct node* ptr = HEAD;
    printf("\nLINKED LIST: \n");

    while (ptr != NULL) {
        printf("%d => ", ptr->data);
        ptr = ptr->next;
    }

    if (size == 0) printf("LINKED LIST IS EMPTY.\n");
    printf("\n");
}

void find() {
    int data;
    printf("Enter the data to search: ");
    scanf("%d", &data);
    if (size == 0) {
        printf("LINKED LIST IS EMPTY.\n");
        return -1;
    }
    int position = 1;
    int found = 0;
    struct node* link = HEAD;
    while (link != NULL) {
        if (link->data == data) {
            printf("Data was found at position: %d", position);
            return;
        }
        link = link->next;
        position++;
    }
    printf("Data is not in the linked list.");
}

void delete() {
    int data;
    printf("Enter the data to delete: ");
    scanf("%d", &data);
    if (size == 0) {
        printf("LINKED LIST IS EMPTY.\n");
        return;
    }
    if (HEAD->data == data) {
        HEAD = HEAD->next;
        return;
    }
    struct node* temp = HEAD;
    while (temp != NULL) {
        if (temp->next->data == data) {
            temp->next = temp->next->next;

```



```

        printf("Data deleted.\n");
        return;
    }
    temp = temp->next;
}
printf("SINCE DATA DOES NOT EXIST, NOTHING WAS REMOVED.\n");
}
void reverse() {
    struct node* previous = NULL;
    struct node* next = NULL;
    struct node* current = HEAD;
    while (current != NULL) {
        next = current->next;
        current->next = previous;
        previous = current;
        current = next;
    }
    HEAD = previous;
}
void printReverse() {
    struct node* last = HEAD;
    struct node* prev = HEAD;
    if (size == 0) return;
    while (last->next != NULL) {
        last = last->next;
    }
    printf("%d -> ", last->data);
    while (last != HEAD) {
        prev = HEAD;
        while (prev->next != last) {
            prev = prev->next;
        }
        last = prev;
        printf("%d -> ", last->data);
    }
    printf("NULL");
}
void sort() {
    struct node* i;
    struct node* j;
    struct node* min;
    for (i = HEAD; i->next != NULL; i = i->next) {
        min = i;
        for (j = i->next; j != NULL; j = j->next) {
            if (j->data < min->data) min = j;
        }
        int temp = i->data;
        i->data = min->data;
        min->data = temp;
    }
}
int main() {
    int ch;
    while (1) {
        printf(
            "\n1. Insert at Beginning\n2. Insert at End\n3. Insert at specific "
            "position\n4. Display\n5. Delete\n6. Reverse "
            "Display\n7. Reverse the linked list\n8. Search\n9. Sort\n"
            "10. Exit\nEnter your "

```

```

        "choice: ");
        scanf("%d", &ch);
        switch (ch) {
        case 1:
            insertBeginning();
            break;
        case 2:
            insertEnd();
            break;
        case 3:
            int data;
            printf("Enter the data to enter: ");
            scanf("%d", &data);
            int pos;
            printf("Enter the pos to input data at: ");
            scanf("%d", &pos);
            insertPos(pos, data);
            break;
        case 4:
            printList();
            break;
        case 5:
            delete ();
            break;
        case 6:
            printReverse();
            break;
        case 7:
            reverse();
            break;
        case 8:
            find();
            break;
        case 9:
            sort();
            break;
        case 10:
            return 0;
        default:
            printf("\nInvalid choice");
        }
    }
}

```

## OUTPUT:

```
1. Insert at Beginning
2.Insert at End
3.Insert at specific postion
4. Display
5. Delete
6. Reverse Display
7. Reverse the linked list
8. Search
9. Sort
10. Exit
Enter your choice: 1
Enter the data to enter: 24
```

```
1. Insert at Beginning
2.Insert at End
3.Insert at specific postion
4. Display
5. Delete
6. Reverse Display
7. Reverse the linked list
8. Search
9. Sort
10. Exit
Enter your choice: 2
Enter the data to enter: 4
```

```
1. Insert at Beginning
2.Insert at End
3.Insert at specific postion
4. Display
5. Delete
6. Reverse Display
7. Reverse the linked list
8. Search
9. Sort
10. Exit
Enter your choice: 4
```

LINKED LIST:

LINKED LIST:

24 => 4 =>

```
1. Insert at Beginning
2.Insert at End
3.Insert at specific postion
4. Display
5. Delete
6. Reverse Display
7. Reverse the linked list
8. Search
9. Sort
10. Exit
Enter your choice: 1
Enter the data to enter: 8
```

```
1. Insert at Beginning
2.Insert at End
3.Insert at specific postion
4. Display
5. Delete
6. Reverse Display
7. Reverse the linked list
8. Search
9. Sort
10. Exit
Enter your choice: 2
Enter the data to enter: 73
```

```
1. Insert at Beginning
2.Insert at End
3.Insert at specific postion
4. Display
5. Delete
6. Reverse Display
7. Reverse the linked list
8. Search
9. Sort
10. Exit
```

```
Enter your choice: 6
73 -> 4 -> 24 -> 8 -> NULL
1. Insert at Beginning
2.Insert at End
3.Insert at specific postion
4. Display
5. Delete
6. Reverse Display
7. Reverse the linked list
8. Search
9. Sort
10. Exit
Enter your choice: 7

1. Insert at Beginning
2.Insert at End
3.Insert at specific postion
4. Display
5. Delete
6. Reverse Display
7. Reverse the linked list
8. Search
9. Sort
10. Exit
Enter your choice: 9

1. Insert at Beginning
2.Insert at End
3.Insert at specific postion
4. Display
5. Delete
6. Reverse Display
7. Reverse the linked list
8. Search
9. Sort
10. Exit
Enter your choice: 10

Process returned 0 (0x0)   execution time : 36.895 s
Press any key to continue.
```

### PROBLEM STATEMENT 13:

Write a C program to write data in a file "myfile.txt" in d:\ and then read and display entire data from file. Also count total alphabets, digits, white space, special characters, and number of lines in the file. Open file using absolute address.

### C SOURCE CODE:

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

int main() {
    FILE *fp;
    char str[100];
    int alphabets = 0, digits = 0, white_space = 0, special_characters = 0,
        lines = 0;

    fp = fopen("d:\\myfile.txt", "w");

    if (fp == NULL) {
        printf("Error in opening file\n");
        exit(1);
    }

    printf("Enter data in file\n");
    do {
        fgets(str, 100, stdin);
        fputs(str, fp);
    } while (str[0] != '\n');
    fclose(fp);

    fp = fopen("d:\\myfile.txt", "r");

    if (fp == NULL) {
        printf("Error in opening file\n");
        exit(1);
    }
    printf("\nData in file is:\n");
    char ch;
    ch = fgetc(fp);
    printf("%c", ch);
    while (ch != EOF) {
        printf("%c", ch);
        if ((ch >= 'a' && ch <= 'z') || (ch >= 'A' && ch <= 'Z')) {
            alphabets++;
        } else if (ch >= '0' && ch <= '9') {
            digits++;
        } else if (ch == ' ') {
            white_space++;
        } else if (ch == '\n') {
            lines++;
        } else {
            special_characters++;
        }
        ch = fgetc(fp);
    }
}
```

```
printf("\n\nTotal alphabets: %d\n", alphabets);
printf("Total digits: %d\n", digits);
printf("Total white spaces: %d\n", white_space);
printf("Total special characters: %d\n", special_characters);
printf("Total lines: %d\n", lines);

fclose(fp);
}
```

OUTPUT:

```
Enter data in file
hell ohstu
ja taet 5345 *
sjdklj 8(

Data in file is:
hell ohstu
ja taet 5345 *
sjdklj 8(

Total alphabets: 21
Total digits: 5
Total white spaces: 5
Total special characters: 2
Total lines: 4

Process returned 0 (0x0)   execution time : 10.075 s
Press any key to continue.
```

#### PROBLEM STATEMENT 14:

Write a C program to write formatted data (Name, department, Eid, Sal, Age) in a file "Emp.dat" in d:\data and then read entire data from file in formatted manner using appropriate method. Open file using absolute address.

#### C SOURCE CODE:

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

struct emp {
    char name[20];
    char dept[20];
    int eid;
    float sal;
    int age;
};

void main() {
    struct emp e;
    FILE *fp;
    char ch;
    fp = fopen("S:\\emp.dat", "w");
    if (fp == NULL) {
        printf("\nFile cannot be opened");
        exit(0);
    }
    do {
        printf("\nEnter name : ");
        scanf("%s", e.name);
        printf("\nEnter department : ");
        scanf("%s", e.dept);
        printf("\nEnter employee id : ");
        scanf("%d", &e.eid);
        printf("\nEnter salary : ");
        scanf("%f", &e.sal);
        printf("\nEnter age : ");
        scanf("%d", &e.age);
        fprintf(fp, "%s %s %d %f %d\n", e.name, e.dept, e.eid, e.sal, e.age);
        printf("\nDo you want to enter more data (y/n) : ");
        ch = getche();
    } while (ch == 'y' || ch == 'Y');
    fclose(fp);
    fp = fopen("S:\\emp.dat", "r");
    if (fp == NULL) {
        printf("\nFile cannot be opened");
        exit(0);
    }
    printf("\n\nName\tDepartment\tEmployee ID\tSalary\tAge");
    while (fscanf(fp, "%s %s %d %f %d", e.name, e.dept, &e.eid, &e.sal, &e.age) !=
        EOF) {
        printf("\n%s\t%s\t%d\t%.2f\t%d", e.name, e.dept, e.eid, e.sal, e.age);
    }
}
```

```
fclose(fp);  
}
```

OUTPUT:

```
Enter name : asyd  
Enter department : phy  
Enter employee id : 123  
Enter salary : 141141  
Enter age : 12  
Do you want to enter more data (y/n) : n  
Name      Department      Employee ID      Salary  Age  
asyd      phy                123             141141.00    12  
Process returned 0 (0x0)   execution time : 24.759 s  
Press any key to continue.
```



### PROBLEM STATEMENT 15:

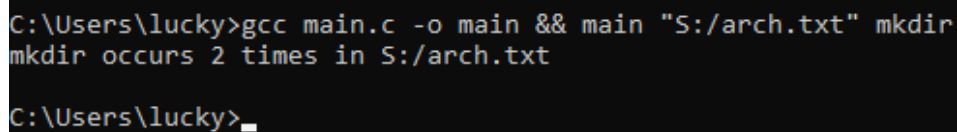
Write a C program to search a string in a file and display the occurrence of the substring in file. Input file name and substring from command prompt as command line argument.

### C SOURCE CODE:

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

int main(int argc, char *argv[]) {
    FILE *fp;
    char buffer[1000];
    int count = 0;
    if (argc != 3) {
        printf("Invalid number of arguments\n");
        return 0;
    }
    fp = fopen(argv[1], "r");
    if (fp == NULL) {
        printf("File not found\n");
        return 0;
    }
    while (fgets(buffer, 1000, fp)) {
        for (int i = 0; i < strlen(buffer); i++) {
            if (strncmp(&buffer[i], argv[2], strlen(argv[2])) == 0) {
                count++;
            }
        }
    }
    printf("%s occurs %d times in %s\n", argv[2], count, argv[1]);
    fclose(fp);
}
```

### OUTPUT:



```
C:\Users\lucky>gcc main.c -o main && main "S:/arch.txt" mkdir
mkdir occurs 2 times in S:/arch.txt
C:\Users\lucky>_
```