**Code :**

#include<stdio.h>

void sort(int arr[], int n){

for(int i = 0; i<n-1; ++i){

int minIndex = i;

int minValue = arr[i];

for(int j = i + 1; j<n; ++j){

if(arr[j] < minValue){

minValue = arr[j];

minIndex = j;

}

}

// swap the current value with min

if(minIndex != i){

arr[i] ^= arr[minIndex];

arr[minIndex] ^= arr[i];

arr[i] ^= arr[minIndex];

}

}

}

int inputArr(int arr[], int n){

for(int i = 0; i<n; ++i){

printf("enter the element of arr[%d]: ", i);

scanf("%d", &arr[i]);

}

}

void print(int arr[], int n){

for(int i = 0; i<n; ++i){

printf("%d ", arr[i]);

}

printf("\n");

}

int main(){

int n;

printf("enter the size of arr: ");

scanf("%d", &n);

int arr[n];

inputArr(arr, n);

sort(arr, n);

int result[n], i = 0, j = n-1, k = 0;

while(i <= j){

result[k++] = arr[i++];

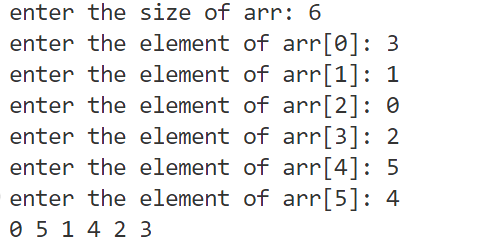
result[k++] = arr[j--];

}

print(result, n);

}

**Output:**



**Code :**

#include<stdio.h>

int isIdentity(int row, int col, int matrix[][col]){

    if(row != col) return 0;

    for(int i = 0; i<row; ++i){

        for(int j = 0; j<col; ++j){

            if(i == j && matrix[i][j] != 1){

                return 0;

            }

            else if(i != j && matrix[i][j] != 0){

                return 0;

            }

        }

    }

    return 1;

}

void inputMatrix(int row, int col, int matrix[][col]){

    for(int i = 0; i<row; ++i){

        for(int j = 0; j<col; ++j){

            scanf("%d", &matrix[i][j]);

        }

    }

}

void printMatrix(int row, int col, int matrix[][col]){

    for(int i = 0; i<row; ++i){

        for(int j = 0; j<col; ++j){

            printf("%d ", matrix[i][j]);

        }

        printf("\n");

    }

}

int main(){

    int row, col;

    printf("Enter row and column: ");

    scanf("%d %d", &row, &col);

    int matrix[row][col];

    inputMatrix(row, col, matrix);

    printMatrix(row, col, matrix);

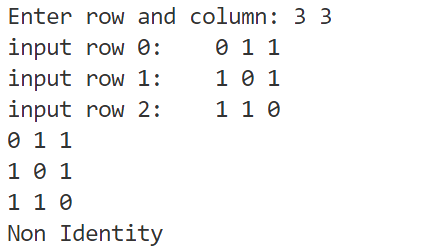
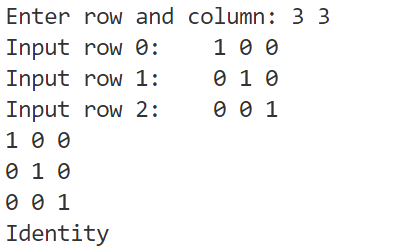
    printf(isIdentity(row, col, matrix)? "Identity" : "Non Identity");

    return 0;

}

**Output :**

Test case 1: Test case 2:

**Code:**

#include<stdio.h>

void inputMatrix(int row, int col, int matrix[][col]){

    for(int i = 0; i<row; ++i){

printf(“input row %d \t”, i);

        for(int j = 0; j<col; ++j){

            scanf("%d", &matrix[i][j]);

        }

    }

}

void printMatrix(int row, int col, int matrix[][col]){

    for(int i = 0; i<row; ++i){

        for(int j = 0; j<col; ++j){

            printf("%d ", matrix[i][j]);

        }

        printf("\n");

    }

}

int main(){

    int r1, c1, r2, c2;

    printf("enter row1 and col1: ");

    scanf("%d %d", &r1, &c1);

    printf("enter row2 and col2: ");

    scanf("%d %d", &r2, &c2);

    if(c1 != r2){

        printf("invalid order");

        return 0;

    }

    int mat1[r1][c1];

    int mat2[r2][c2];

    inputMatrix(r1, c1, mat1);

    inputMatrix(r2, c2, mat2);

    printf("Matrix 1: \n");

    printMatrix(r1, c1, mat1);

    printf("Matrix 2: \n");

    printMatrix(r2, c2, mat2);

    // multiplication of matrix

    int result[r1][c2];

    for(int i = 0; i<r1; ++i){

        for(int j = 0; j<c2; ++j){

            result[i][j] = 0;

            for(int k = 0; k<r2; ++k){

                result[i][j] += mat1[i][k] \* mat2[k][j];

            }

        }

    }

    printf("Result: \n");

    printMatrix(r1, c2, result);

    int transpose[c2][r1];

    for(int i = 0; i<c2; ++i){

        for(int j = 0; j<r1; ++j){

            transpose[i][j] = result[j][i];

        }

    }

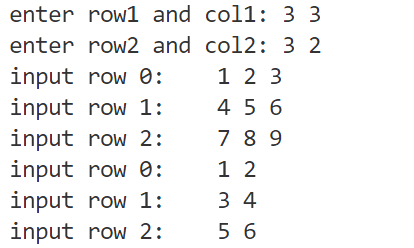
    printf("Transpose: \n");

    printMatrix(r1, c2, transpose);

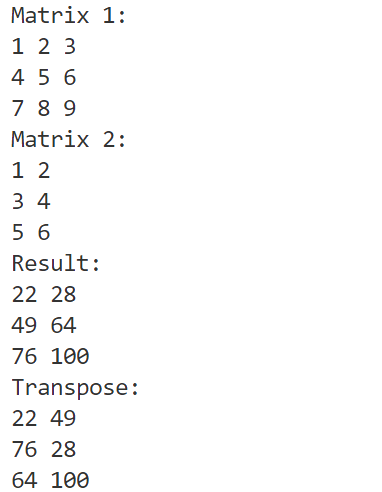
    return 0;

}

**Input :**



**Output:**



**Code:**

#include <stdio.h>

void printArr(int arr[], int n) {

    for (int i = 0; i < n; ++i) {

        printf("%d ", arr[i]);

    }

    printf("\n");

}

struct Pair{

    int first, second;

} typedef Pair;

void insert(Pair freqArr[], int \*n, int value){

    // increase the frequency of existing element

    for(int i = 0; i<\*n; ++i){

        if(freqArr[i].first == value){

            freqArr[i].second++;

            return;

        }

    }

    // add a new element

    freqArr[\*n].first = value;

    freqArr[\*n].second = 1;

    (\*n)++;

}

int main() {

    int arr[] = {2, 3, 4, 5, 4, 3, 1, 7, 8, 9, 8, 5, 1, 4, 5};

    int n = sizeof(arr) / sizeof(int);

    Pair freqArr[n];

    int size = 0;

    for(int i = 0; i<n; ++i){

        insert(freqArr, &size, arr[i]);

    }

    printf("\nUnique numbers are: ");

    for(int i = 0; i<size; ++i){

        if(freqArr[i].second == 1){

            printf("%d ", freqArr[i].first);

        }

    }

    printf("\nDuplicate numbers are: ");

    for(int i = 0; i<size; ++i){

        if(freqArr[i].second > 1){

            printf("%d ", freqArr[i].first);

        }

    }

    printf("\nOccurrence of each numbers are:\n");

    for(int i = 0; i<size; ++i){

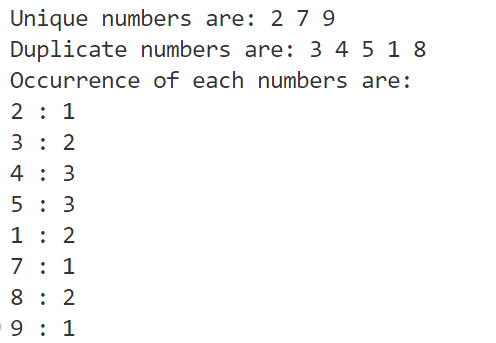
        printf("%d : %d\n", freqArr[i].first, freqArr[i].second);

    }

    return 0;

}

**Output**



**Code:**

#include<stdio.h>

void selectionSort(int arr[], int n){

    for(int i = 0; i<n-1; ++i){

        int minIndex = i;

        int minValue = arr[i];

        for(int j = i + 1; j<n; ++j){

            if(arr[j] < minValue){

                minValue = arr[j];

                minIndex = j;

            }

        }

        // swap the current value with min

        if(minIndex != i){

            arr[i] ^= arr[minIndex];

            arr[minIndex] ^= arr[i];

            arr[i] ^= arr[minIndex];

        }

    }

}

int inputArr(int arr[], int n){

    for(int i = 0; i<n; ++i){

        printf("enter the element of arr[%d]: ", i);

        scanf("%d", &arr[i]);

    }

}

void printArr(int arr[], int n){

    for(int i = 0; i<n; ++i){

        printf("%d ", arr[i]);

    }

    printf("\n");

}

int main(){

    int n;

    printf("Enter the size of arr: ");

    scanf("%d", &n);

    int arr[n];

    inputArr(arr, n);

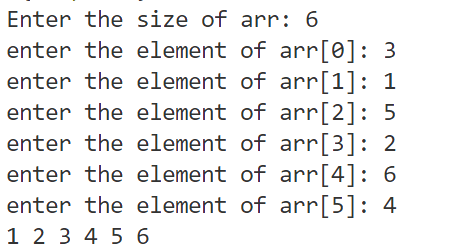
    selectionSort(arr, n);

    printArr(arr, n);

    return 0;

}

**Output:**

****

**Code:**

#include<stdio.h>

int binarySearch(int arr[], int low, int high, int key){

    if(low > high) return -1;

    int mid = (low + high) / 2;

    if(arr[mid] == key) return mid;

    if(key < arr[mid]){

        return binarySearch(arr, low, mid-1, key);

    }

    else{

        return binarySearch(arr, mid+1, high, key);

    }

}

void sort(int arr[], int n){

    for(int i = 0; i<n-1; ++i){

        int minIndex = i;

        int minValue = arr[i];

        for(int j = i + 1; j<n; ++j){

            if(arr[j] < minValue){

                minValue = arr[j];

                minIndex = j;

            }

        }

        // swap the current value with min

        if(minIndex != i){

            arr[i] ^= arr[minIndex];

            arr[minIndex] ^= arr[i];

            arr[i] ^= arr[minIndex];

        }

    }

}

int inputArr(int arr[], int n){

    for(int i = 0; i<n; ++i){

        printf("enter the element of arr[%d]: ", i);

        scanf("%d", &arr[i]);

    }

}

void printArr(int arr[], int n){

    for(int i = 0; i<n; ++i){

        printf("%d ", arr[i]);

    }

    printf("\n");

}

int main(){

    int n, key;

    printf("Enter the size of arr: ");

    scanf("%d", &n);

    int arr[n];

    inputArr(arr, n);

    printf("Enter the key: ");

    scanf("%d", &key);

    sort(arr, n);

    printArr(arr, n);

    int result = binarySearch(arr, 0, n, key);

    if(result != -1){

        printf("key found at %d\n", result);

    }

    else{

        printf("key not found\n");

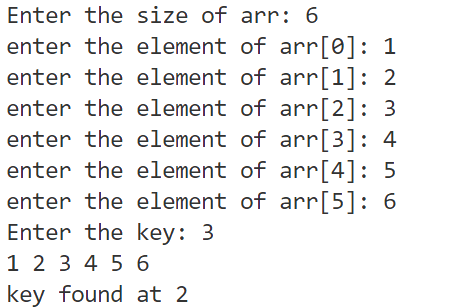
    }

    return 0;

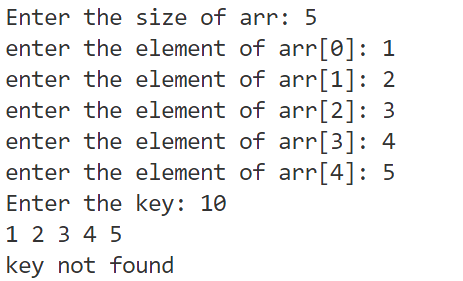
}

**Output:**

Test Case 1:



Test Case 2:



**Code:**

#include<stdio.h>

#include<string.h>

int main(){

    int lastIndex[256];

    for(int i = 0; i<256; ++i){

        lastIndex[i] = -1;

    }

    char str[100];

    printf("Enter a string: ");

    gets(str);

    int maxLen = 0;

    int i = 0;

    for(int j = 0; j<strlen(str); ++j){

        if(lastIndex[str[j]] >= i){

            i = lastIndex[str[j]] + 1;

        }

        lastIndex[str[j]] = j;

        int len = j - i + 1;

        if(len > maxLen){

            maxLen = len;

        }

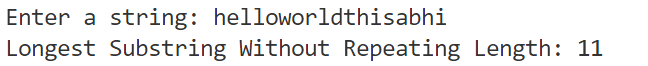
    }

    printf("Longest Substring Without Repeating Length: %d", maxLen);

    return 0;

}

**Output:**



**Code:**

#include<stdio.h>

struct Employee{

    char name[50];

    char department[50];

    int id;

    long int salary;

} typedef Employee;

void printRecord(Employee record[], int n){

    for(int i = 0; i<n; ++i){

        printf("%d %10s %10s %10ld\n",

        record[i].id, record[i].name, record[i].department, record[i].salary);

    }

}

void inputRecord(Employee record[], int n){

    for(int i = 0; i<n; ++i){

        printf("enter the id, name, department, salary: ");

        scanf("%d %s %s %ld",

        &record[i].id, &record[i].name, &record[i].department, &record[i].salary);

    }

}

int main(){

    int n;

    printf("enter the size of record: ");

    scanf("%d", &n);

    Employee record[n];

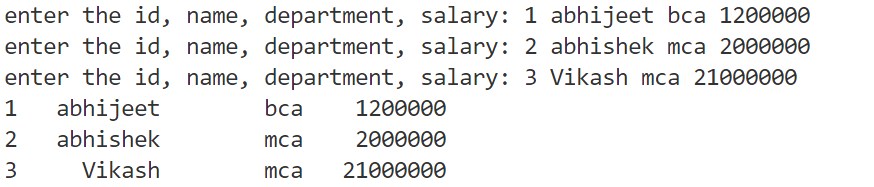
    inputRecord(record, n);

    printRecord(record, n);

    return 0;

}

**Output:**



**Code:**

#include<stdio.h>

#include<string.h>

#define MAX\_STUDENTS 100

struct Student

{

    char name[20];

    char course[10];

    int roll;

} typedef Student;

void insert(Student records[], int \*n){

    if(\*n >= MAX\_STUDENTS){

        printf("Record is full can't add more students.\n");

        return;

    }

    printf("Enter roll no: ");

    scanf("%d", &records[\*n].roll);

    printf("Enter Name: ");

    scanf(" %[^\n]", records[\*n].name);

    printf("Enter Course: ");

    scanf(" %[^\n]", records[\*n].course);

    (\*n)++;

    printf("one record inserted successfully\n");

}

void display(Student records[], int n){

    for(int i = 0; i<n; ++i){

        printf("%d %s %s\n", records[i].roll, records[i].name, records[i].course);

    }

}

void delete(Student records[], int \*n){

    int roll;

    printf("Enter roll no of student to delete: ");

    scanf("%d", &roll);

    int isFound = 0, i , j;

    for(i = 0; i<\*n; ++i){

        if(records[i].roll == roll){

            isFound = 1;

            break;

        }

    }

    if(!isFound){

        printf("student with %d roll not no found\n");

        return;

    }

    for(j = i; j < (\*n) - 1; j++){

        records[j] = records[j+1];

    }

    (\*n)--;

    printf("student with %d roll no deleted.\n");

}

void search(Student records[], int n){

    int roll;

    printf("Enter roll no of student to delete: ");

    scanf("%d", &roll);

    int isFound = 0, i , j;

    for(i = 0; i<n; ++i){

        if(records[i].roll == roll){

            isFound = 1;

            break;

        }

    }

    if(!isFound){

        printf("student with %d roll not no found\n");

        return;

    }

    printf("%d %s %s\n", records[i].roll, records[i].name, records[i].course);

}

int main(){

    Student records[MAX\_STUDENTS];

    int n = 0;

    int choice;

    do{

        printf("\nMenu:\n");

        printf("1. Insert\n");

        printf("2. Display\n");

        printf("3. Delete\n");

        printf("4. Search\n");

        printf("5. Exit\n");

        printf("enter your choice: ");

        scanf("%d", &choice);

        switch (choice)

        {

            case 1: insert(records, &n); break;

            case 2: display(records, n); break;

            case 3: delete(records, &n); break;

            case 4: search(records, n);  break;

            default: printf("Invalid choice\n"); break;

        }

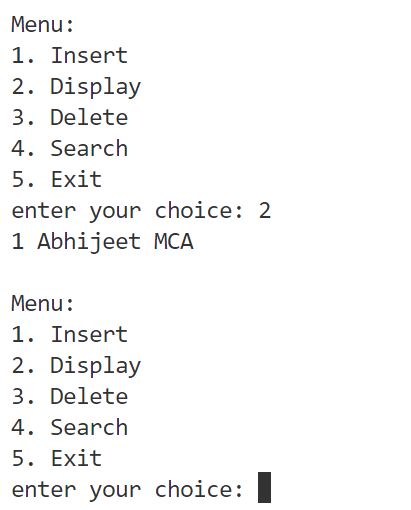
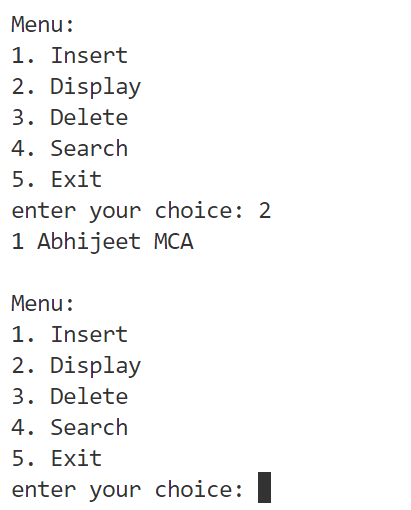
    }while(choice != 5);

    return 0;

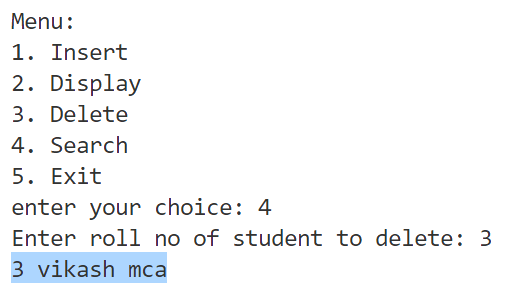
}

**Output:**

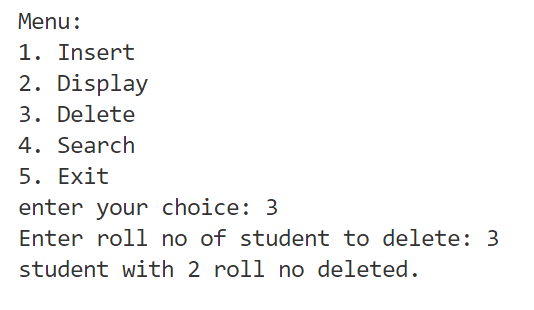
**Insert Display**

**Search**



**Delete**



**Code:**

#include <stdio.h>

#include <stdlib.h>

struct Node

{

    int data;

    struct Node \*next;

};

typedef struct Node Node;

Node \*getNode(int data)

{

    Node \*node = (Node \*)malloc(sizeof(Node));

    node->data = data;

    node->next = NULL;

    return node;

}

void insertAtBegin(Node \*\*head, int data)

{

    Node \*newNode = getNode(data);

    if (!\*head)

    {

        \*head = newNode;

        return;

    }

    newNode->next = \*head;

    \*head = newNode;

}

void insertAtEnd(Node \*\*head, int data)

{

    if (!\*head)

    {

        insertAtBegin(head, data);

        return;

    }

    Node \*newnode = getNode(data);

    Node \*curr;

    for (curr = \*head; curr->next; curr = curr->next)

        ;

    curr->next = newnode;

}

void insertAt(Node \*\*head, int data, int pos)

{

    if (pos <= 0)

    {

        insertAtBegin(head, data);

        return;

    }

    Node \*newnode = getNode(data);

    Node \*temp = \*head;

    for (int i = 1; i < pos && temp->next; ++i, temp = temp->next)

        ;

    newnode->next = temp->next;

    temp->next = newnode;

}

void reverse(Node \*\*head)

{

    if (!\*head || !(\*head)->next)

        return;

    Node \*prev = NULL;

    Node \*curr = \*head;

    Node \*next = (\*head)->next;

    while (curr)

    {

        next = curr->next;

        curr->next = prev;

        prev = curr;

        curr = next;

    }

    \*head = prev;

}

void reverseDisplay(Node \*head)

{

    Node \*lastVisited = NULL;

    while (lastVisited != head)

    {

        Node \*curr;

        for (curr = head; curr->next != lastVisited; curr = curr->next)

            ;

        printf("%d ", curr->data);

        lastVisited = curr;

    }

    printf("\n");

}

void sort(Node \*head)

{

    for (Node \*i = head; i->next; i = i->next)

    {

        Node \*min = i;

        for (Node \*j = i->next; j; j = j->next)

        {

            if (j->data < min->data)

            {

                min = j;

            }

        }

        if (min != i)

        {

            i->data ^= min->data;

            min->data ^= i->data;

            i->data ^= min->data;

        }

    }

}

int search(Node \*head, int key)

{

    int i = 0;

    while (head)

    {

        if (head->data == key)

            return i;

        head = head->next;

        i++;

    }

    return -1;

}

void delete(Node \*\*head, int key)

{

    Node \*toDelete;

    // Delete at Beginning

    if (\*head && (\*head)->data == key)

    {

        toDelete = \*head;

        \*head = (\*head)->next;

        free(toDelete);

        return;

    }

    Node \*curr = \*head;

    while (curr && curr->next && curr->next->data != key)

    {

        curr = curr->next;

    }

    // If the key was not found

    if (!curr || !curr->next)

    {

        return;

    }

    toDelete = curr->next;

    curr->next = curr->next->next;

    free(toDelete);

}

void display(Node \*head)

{

    while (head)

    {

        printf("%d -> ", head->data);

        head = head->next;

    }

    printf("NULL\n");

}

int main()

{

    Node \*head = NULL;

    printf("1. Insert at Beginning\n");

    printf("2. Insert at End\n");

    printf("3. Insert at Specific Position\n");

    printf("4. Display\n");

    printf("5. Delete\n");

    printf("6. Reverse Display\n");

    printf("7. Reverse the Linked List\n");

    printf("8. Search\n");

    printf("9. Sort(using selection sort)\n");

    int data, pos, key;

    while (1)

    {

        int choice;

        printf("Enter your choice: ");

        scanf("%d", &choice);

        switch (choice)

        {

        case 1:

            printf("Enter the data: ");

            scanf("%d", &data);

            insertAtBegin(&head, data);

            break;

        case 2:

            printf("Enter the data: ");

            scanf("%d", &data);

            insertAtEnd(&head, data);

            break;

        case 3:

            printf("Enter the data and position: ");

            scanf("%d %d", &data, &pos);

            insertAt(&head, data, pos);

            break;

        case 4:

            display(head);

            break;

        case 5:

            printf("Enter the key: ");

            scanf("%d", &key);

            delete (&head, key);

            break;

        case 6:

            reverseDisplay(head);

            break;

        case 7:

            reverse(&head);

            break;

        case 8:

            printf("Enter the key: ");

            scanf("%d", &key);

            printf(search(head, key) != -1 ? "Found\n" : "Not Found\n");

            break;

        case 9:

            sort(head);

            break;

        default:

            printf("Invalid choice\n");

            break;

        }

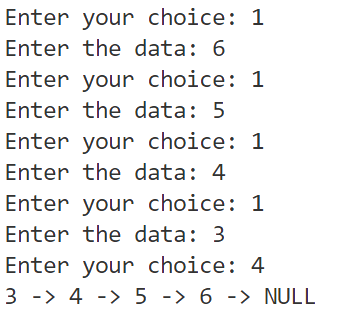
    }

    return 0;

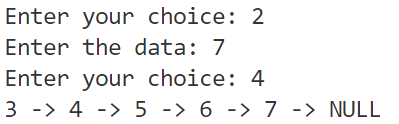
}

**Output:**

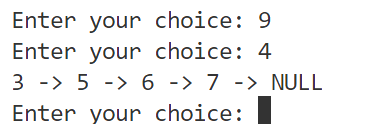
**Input at beginning**



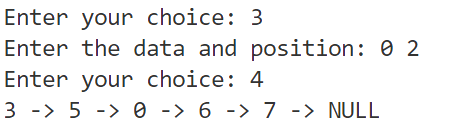
**Input at ending**

****

**Sort**



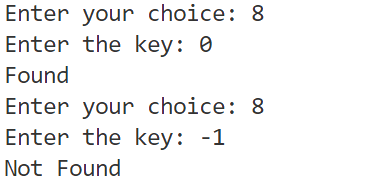
**Insert at Specific Position**

****

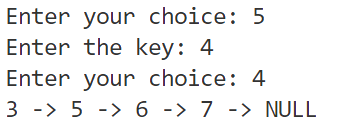
**Reverse and Display**

****

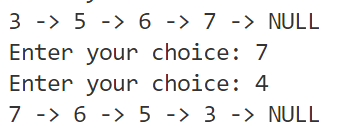
**Search**



**Delete**

****

**Reverse**

****

**Code:**

#include <stdio.h>

#include <ctype.h>

int main() {

    FILE \*file;

    char ch;

    int alphabets = 0, digits = 0, whitespaces = 0, specialchars = 0, lines = 0;

    // Write to the file

    file = fopen("C:\\users\\abhij\\desktop\\Data.txt", "w");

    if (file == NULL) {

        perror("Unable to open file in write mode");

        return 1;

    }

    fputs("Hello World!\nThis is a test file with 123 numbers.", file);

    fclose(file);

    // Read from the file and count characters

    file = fopen("C:\\users\\abhij\\desktop\\Data.txt", "r");

    if (file == NULL) {

        perror("Unable to open file in read mode");

        return 1;

    }

    while ((ch = fgetc(file)) != EOF) {

        if (isalpha(ch))

            alphabets++;

        else if (isdigit(ch))

            digits++;

        else if (isspace(ch)) {

            whitespaces++;

            if (ch == '\n')

                lines++;

        } else

            specialchars++;

    }

    fclose(file);

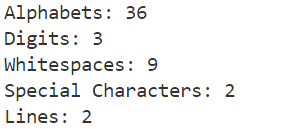
    printf("Alphabets: %d\nDigits: %d\nWhitespaces: %d\nSpecial Characters: %d\nLines: %d\n",

           alphabets, digits, whitespaces, specialchars, lines + 1);

    return 0;

}

**Output:**



#include <stdio.h>

typedef struct {

    char name[50];

    char department[50];

    int eid;

    float salary;

    int age;

} Employee;

int main() {

    FILE \*file;

    Employee emp = {"Gagandeep", "Engineering", 101, 60000.00, 30};

    // Write to the file

    file = fopen("C:\\users\\abhij\\desktop\\Emp.dat", "wb");

    if (file == NULL) {

        printf("Unable to open file in write mode\n");

        return 0;

    }

    fwrite(&emp, sizeof(Employee), 1, file);

    fclose(file);

    // Read from the file

    file = fopen("C:\\users\\abhij\\desktop\\Emp.dat", "rb");

    if (file == NULL) {

        printf("Unable to open file in read mode\n");

        return 0;

    }

    fread(&emp, sizeof(Employee), 1, file);

    fclose(file);

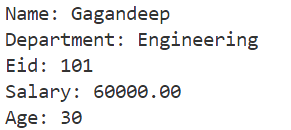
    printf("Name: %s\nDepartment: %s\nEid: %d\nSalary: %.2f\nAge: %d\n",

           emp.name, emp.department, emp.eid, emp.salary, emp.age);

    return 0;

}

**Output**



#include <stdio.h>

#include <stdlib.h>

#include <string.h>

int main(int argc, char \*argv[]) {

    if (argc != 3) {

        printf("Usage: %s <filename> <substring>\n", argv[0]);

        return 1;

    }

    FILE \*file;

    char \*filename = argv[1];

    char \*substring = argv[2];

    char line[256];

    int count = 0;

    // Open file

    file = fopen(filename, "r");

    if (file == NULL) {

        printf("Unable to open file %s\n", filename);

        return 1;

    }

    // Search for the substring

    while (fgets(line, sizeof(line), file)) {

        if (strstr(line, substring) != NULL)

            count++;

    }

    fclose(file);

    printf("The substring '%s' occurs %d times in the file '%s'.\n",

           substring, count, filename);

    return 0;

}

**Output:**

