

Array operations-insertion,deletion,searching:

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

```
int main()
```

```
{
```

```
    int a[50],i,n,p,val,num=0;
```

```
    printf("enter the choose you want: ");
```

```
    scanf("%d",&num);
```

```
    switch(num)
```

```
    {
```

```
        case 1:
```

```
            printf("enter the no .of elemnts in the array:");
```

```
            scanf("%d",&n);
```

```
            for(i=0;i<n;i++)
```

```
            {
```

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

```
            }
```

```
            printf("enter the position to insert an element:");
```

```
            scanf("%d",&p);
```

```
            printf("enter the value to insert");
```

```
            scanf("%d",&val);
```

```
            for(i=i-1;i>=p-1;i--)
```

```
                a[i+1]=a[i];
```

```
                a[p-1]= val;
```

```
            printf("the final array of elements are:");
```

```
            for(i=0;i<=n;i++)
```

```
                printf("%d\n",a[i]);
```

```
            break;
```

```
        case 2:
```

```
int arr[50];
```

```
int pos, i, num;
```

```
printf (" \n Enter the number of elements in an array: \n ");
```

```
scanf (" %d", &num);
```

```
printf (" \n Enter %d elements in array: \n ", num);
```

```
for (i = 0; i < num; i++ )
```

```
{
```

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

```
}
```

```
printf( " Define the position of the array element where you want to delete: \n ");
```

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

```
if (pos >= num+1)
```

```
{
```

```
    printf (" \n Deletion is not possible in the array.");
```

```
}
```

```
else
```

```
{
```

```
    for (i = pos - 1; i < num -1; i++)
```

```
    {
```

```
        arr[i] = arr[i+1];
```

```
    }
```

```
    printf (" \n The resultant array is: \n");
```

```
    for (i = 0; i < num - 1; i++)
```

```
    {
```

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

```
    }
```

```
}
```

```
break;
```

```
case 3:
```

```
#define MAX_SIZE 100
```

```
{
```

```
    int arr[MAX_SIZE];
```

```
int size, i, toSearch, found;

printf("Enter size of array: ");

scanf("%d", &size);

printf("Enter elements in array: ");

for(i=0; i<size; i++)

{

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

}


printf("\nEnter element to search: ");

scanf("%d", &toSearch);

found = 0;


for(i=0; i<size; i++)

{

    if(arr[i] == toSearch)

    {

        found = 1;

        break;

    }

}

if(found==1)

{

    printf("\n%d is found at position %d",toSearch,i+1);

}

else

{

    printf("\n%d is not found in the array",toSearch);

}


return 0;
```

```

}

break;

default:

    printf("you entered invalid option");

    break;

}

return 0;

}

```

```

C:\Users\sivas\OneDrive\Documents\ds- array oprations using switch case.exe
enter the choose you want: 1
enter the no .of elemnts in the array:5
1
2
3
4
5
enter the position to insert an element:3
enter the value to insert8
the final array of elements are:1
2
8
3
4
5
-----
Process exited after 13.25 seconds with return value 0
Press any key to continue . . .

```

Searching a number using linear search:

```

#include<stdio.h>

int main()
{
    int a[20],i,x,n;

    printf("enter the number of elements you want: ");

    scanf("%d",&n);

    printf("Enter array elements: ");

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

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

    printf("Enter element to search:");

```

```

scanf("%d",&x);

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

    if(a[i]==x)

        break;

if(i<n)

    printf("Element found at index %d",i);

else

    printf("Element not found");

return 0;

}

```

```

C:\Users\sivas\OneDrive\Documents\ds-searching element using linear search.exe
enter the number of elements you want: 5
Enter array elements: 1
2
3
4
5
Enter element to search:3
Element found at index 2
-----
Process exited after 10.14 seconds with return value 0
Press any key to continue . . .

```

Searching number using binary search:

```

#include<stdio.h>

int main()

{

int c,first,last,middle,n,search,array[100];

```

```
printf("Enter number of elements you want: \n");
scanf("%d",&n);
printf("Enter %d integers\n",n);
for (c=0 ;c<n;c++)
scanf("%d",&array[c]);
printf("Enter value to find\n");
scanf("%d",&search);
first = 0;
last = n-1;
middle = (first+last)/2;
while(first<=last)
{
    if (array[middle]<search)
        first = middle + 1;
    else if (array[middle] == search )
    {
        printf("%d found at location %d.\n", search, middle+1);
        break;
    }
    else
        last = middle-1;
        middle = (first+last)/2;
}
if ( first>last )
    printf("Not found %d is not present in the list\n", search);
return 0;
}
```

```
C:\Users\sivas\OneDrive\Documents\ds-searching a number using binary search.exe
Enter number of elements you want:
5
Enter 5 integers
5
6
7
8
9
Enter value to find
1
Not found 1 is not present in the list
-----
Process exited after 12.39 seconds with return value 0
Press any key to continue . . .
```

Inserting element at begin in single link list:

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

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

```
struct Node
```

```
{
```

```
    int data;
```

```
    struct Node *next;
```

```
};
```

```
void insertStart(struct Node** head, int data){
```

```
    struct Node* newNode = (struct Node*) malloc(sizeof(struct Node));
```

```
    newNode->data = data;
```

```
    newNode->next = *head;
```

```
    *head = newNode;
```

```
}
```

```
void insertLast(struct Node** head, int data){
```

```
    struct Node* newNode = (struct Node*) malloc(sizeof(struct Node));
```

```
    newNode->data = data;
```

```

newNode->next = NULL;
if(*head==NULL)
{
    *head = newNode;
    return;
}
struct Node* temp = *head;
while(temp->next!=NULL)
    temp = temp->next;
temp->next = newNode;
}
void display(struct Node* node)
{
    while(node!=NULL){
        printf("%d ",node->data);
        node = node->next;
    }
    printf("\n");
}
int main()
{
    struct Node* head = NULL;
    struct Node* node2 = NULL;
    struct Node* node3 = NULL;
    struct Node* node4 = NULL;

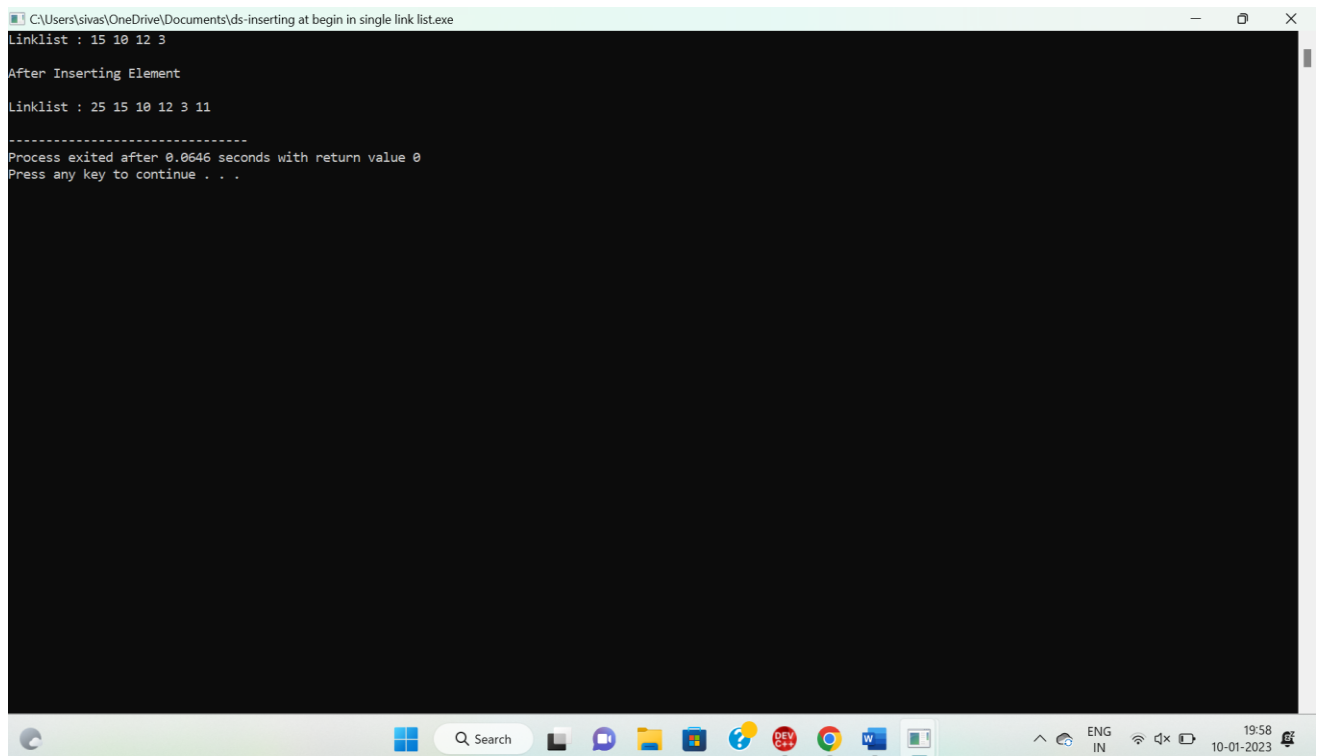
    head = (struct Node*)malloc(sizeof(struct Node));
    node2 = (struct Node*)malloc(sizeof(struct Node));
    node3 = (struct Node*)malloc(sizeof(struct Node));
    node4 = (struct Node*)malloc(sizeof(struct Node));

    head->data = 15;
    head->next = node2;

```



```
node2->data = 10;
node2->next = node3;
node3->data = 12;
node3->next = node4;
node4->data = 3;
node4->next = NULL;
printf("Linklist : ");
display(head);
insertStart(&head,25);
insertLast(&head,11);
printf("\nAfter Inserting Element\n");
printf("\nLinklist : ");
display(head);
return 0;
}
```



The screenshot shows a Windows command prompt window titled "C:\Users\sivas\OneDrive\Documents\ds-inserting at begin in single link list.exe". The output of the program is as follows:

```
Linklist : 15 10 12 3
After Inserting Element
Linklist : 25 15 10 12 3 11
-----
Process exited after 0.0646 seconds with return value 0
Press any key to continue . . .
```

The Windows taskbar at the bottom shows the Start button, a search bar, and several application icons including File Explorer, Microsoft Edge, and Word. The system tray on the right indicates the language is set to "ENG IN", and the date and time are "10-01-2023" and "19:58" respectively.

Inserting a element at end in single link list:

```
#include<stdio.h>

#include<stdlib.h>

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

void insertStart(struct Node** head, int data){

    struct Node* newNode = (struct Node*) malloc(sizeof(struct Node));

    newNode->data = data;
    newNode->next = *head;
    *head = newNode;
}

void insertLast(struct Node** head, int data){
    struct Node* newNode = (struct Node*) malloc(sizeof(struct Node));

    newNode->data = data;
    newNode->next = NULL;
    if(*head==NULL){
        *head = newNode;
        return;
    }
    struct Node* temp = *head;

    while(temp->next!=NULL)
        temp = temp->next;

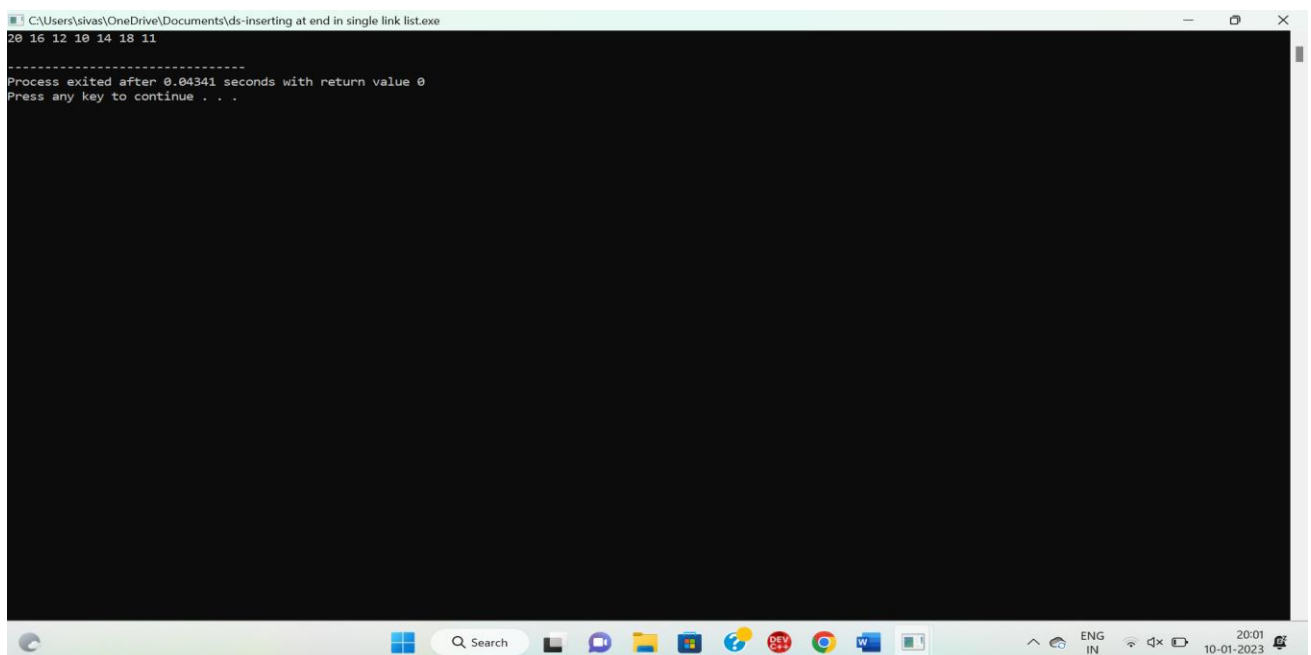
    temp->next = newNode;
}
```

```

void display(struct Node* node){
    while(node!=NULL){
        printf("%d ",node->data);
        node = node->next;
    }
    printf("\n");
}

int main()
{
    struct Node* head = NULL;
    insertStart(&head,12);
    insertStart(&head,16);
    insertStart(&head,20);
    insertLast(&head,10);
    insertLast(&head,14);
    insertLast(&head,18);
    insertLast(&head,11);
    display(head);
    return 0;
}

```



```

C:\Users\sivas\OneDrive\Documents\ds-inserting at end in single link list.exe
20 16 12 10 14 18 11
-----
Process exited after 0.04341 seconds with return value 0
Press any key to continue . . .

```

Inserting element at any position in single link list:

```
#include<stdio.h>

#include<stdlib.h>

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

int calcSize(struct Node* node){
    int size=0;

    while(node!=NULL){
        node = node->next;
        size++;
    }
    return size;
}

void insertPosition(int pos, int data, struct Node** head)
{
    int size = calcSize(*head);
    if(pos < 1 || size < pos)
    {
        printf("Can't insert, %d is not a valid position\n",pos);
    }
    else
    {
        struct Node* temp = *head;
        struct Node* newNode = (struct Node*)
        malloc(sizeof(struct Node));
        newNode->data = data;
        newNode->next = NULL;
```

```

while(--pos)
{
    temp=temp->next;
}
newNode->next= temp->next;
temp->next = newNode;
}
}

void insertStart(struct Node** head, int data){

    struct Node* newNode = (struct Node*) malloc(sizeof(struct Node));
    newNode->data = data;
    newNode->next = *head;
    *head = newNode;
}

void insertLast(struct Node** head, int data){

    struct Node* newNode = (struct Node*) malloc(sizeof(struct Node));
    newNode->data = data;
    newNode->next = NULL;
    if(*head==NULL){
        *head = newNode;
        return;
    }
    struct Node* temp = *head;
    while(temp->next!=NULL)
        temp = temp->next;
    temp->next = newNode;
}

void display(struct Node* node){

```

```

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

int main()
{
    struct Node* head = NULL;
    insertStart(&head,12);
    insertStart(&head,16);
    insertStart(&head,20);
    insertLast(&head,10);
    insertLast(&head,14);
    insertLast(&head,18);
    insertLast(&head,11);
    insertPosition(3,25,&head);
    display(head);
    return 0;
}

```

```

C:\Users\sivas\OneDrive\Documents\ds- inserting at any position in single link list.exe
20 16 12 25 10 14 18 11
-----
Process exited after 0.04705 seconds with return value 0
Press any key to continue . . .

```

Delete at begin in single link list:

```
#include <stdio.h>

#include <stdlib.h>

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

void createList(int n);
void deleteFirstNode();
void displayList();

int main()
{
    int n, choice;

    printf("Enter the total number of nodes: ");
    scanf("%d", &n);
    createList(n);
    printf("\nData in the list \n");
    displayList();

    printf("\nPress 1 to delete first node: ");
    scanf("%d", &choice);
    if(choice == 1)
        deleteFirstNode();
    printf("\nData in the list \n");
    displayList();
    return 0;
}

void createList(int n)
{
    struct node *newNode, *temp;
    int data, i;
```

```

head = (struct node *)malloc(sizeof(struct node));
if(head == NULL)
{
    printf("Unable to allocate memory.");
}
else
{
    printf("Enter the data of node 1: ");
    scanf("%d", &data);
    head->data = data;
    head->next = NULL;
    temp = head;
    for(i=2; i<=n; i++)
    {
        newNode = (struct node *)malloc(sizeof(struct node));
        if(newNode == NULL)
        {
            printf("Unable to allocate memory.");
            break;
        }
        else
        {
            printf("Enter the data of node %d: ", i);
            scanf("%d", &data);
            newNode->data = data;
            newNode->next = NULL;
            temp->next = newNode;
            temp = temp->next;
        }
    }
    printf("SINGLY LINKED LIST CREATED SUCCESSFULLY\n");
}

```



```

    }
}

void deleteFirstNode()
{
    struct node *toDelete;
    if(head == NULL)
    {
        printf("List is already empty.");
    }
    else
    {
        toDelete = head;
        head = head->next;
        printf("\nData deleted = %d\n", toDelete->data);
        free(toDelete);
        printf("SUCCESSFULLY DELETED FIRST NODE FROM LIST\n");
    }
}

void displayList()
{
    struct node *temp;
    if(head == NULL)
    {
        printf("List is empty.");
    }
    else
    {
        temp = head;
        while(temp != NULL)
        {
            printf("Data = %d\n", temp->data);

```

```

        temp = temp->next;
    }
}
}

```

The screenshot shows a Windows command prompt window titled "C:\Users\sivas\OneDrive\Documents\ds- delete at begin in single link list.exe". The program prompts the user to enter the total number of nodes (5) and then the data for each node (5, 6, 7, 8, 9). It then displays the data in the list. The user is prompted to press 1 to delete the first node, and the program confirms the deletion of the first node (5). It then displays the updated list (6, 7, 8, 9). The program exits after 8.359 seconds with a return value of 0. The Windows taskbar at the bottom shows the date and time as 10-01-2023, 20:17.

```

C:\Users\sivas\OneDrive\Documents\ds- delete at begin in single link list.exe
Enter the total number of nodes: 5
Enter the data of node 1: 5
Enter the data of node 2: 6
Enter the data of node 3: 7
Enter the data of node 4: 8
Enter the data of node 5: 9
SINGLY LINKED LIST CREATED SUCCESSFULLY

Data in the list
Data = 5
Data = 6
Data = 7
Data = 8
Data = 9

Press 1 to delete first node: 1

Data deleted = 5
SUCCESSFULLY DELETED FIRST NODE FROM LIST

Data in the list
Data = 6
Data = 7
Data = 8
Data = 9

-----
Process exited after 8.359 seconds with return value 0
Press any key to continue . . .

```

Delete at end in single link list:

```

#include <stdio.h>

#include <stdlib.h>

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

void createList(int n);

void deleteLastNode();

void displayList();

int main()
{
    int n, choice;

    printf("Enter the total number of nodes: ");

```

```

scanf("%d", &n);
createList(n);
printf("\nData in the list \n");
displayList();
printf("\nPress 1 to delete last node: ");
scanf("%d", &choice);
if(choice == 1)
    deleteLastNode();
printf("\nData in the list \n");
displayList();
return 0;
}

void createList(int n)
{
    struct node *newNode, *temp;
    int data, i;
    head = (struct node *)malloc(sizeof(struct node));
    if(head == NULL)
    {
        printf("Unable to allocate memory.");
    }
    else
    {
        printf("Enter the data of node 1: ");
        scanf("%d", &data);
        head->data = data;
        head->next = NULL;
        temp = head;
        for(i=2; i<=n; i++)
        {
            newNode = (struct node *)malloc(sizeof(struct node));

```

```

        if(newNode == NULL)
        {
            printf("Unable to allocate memory.");
            break;
        }
        else
        {
            printf("Enter the data of node %d: ", i);
            scanf("%d", &data);

            newNode->data = data;
            newNode->next = NULL;

            temp->next = newNode;
            temp = temp->next;
        }
    }

    printf("SINGLY LINKED LIST CREATED SUCCESSFULLY\n");
}

void deleteLastNode()
{
    struct node *toDelete, *secondLastNode;

    if(head == NULL)
    {
        printf("List is already empty.");
    }
    else
    {

```

```

toDelete = head;
secondLastNode = head;
while(toDelete->next != NULL)
{
    secondLastNode = toDelete;
    toDelete = toDelete->next;
}
if(toDelete == head)
{
    head = NULL;
}
else
{
    secondLastNode->next = NULL;
}
free(toDelete);
printf("SUCCESSFULLY DELETED LAST NODE OF LIST\n");
}
}

void displayList()
{
    struct node *temp;
    if(head == NULL)
    {
        printf("List is empty.");
    }
    else
    {
        temp = head;
        while(temp != NULL)
        {

```

```

        printf("Data = %d\n", temp->data);

        temp = temp->next;

    }

}
}

```

The screenshot shows a Windows command prompt window titled "C:\Users\sivas\OneDrive\Documents\ds-delete at end in single link list.exe". The program prompts the user to enter the total number of nodes (5) and then the data for each node (1, 3, 5, 7, 9). It then displays the data in the list. After pressing 1 to delete the last node, it shows the updated list (1, 3, 5, 7) and the message "SUCCESSFULLY DELETED LAST NODE OF LIST". The process exits after 11.61 seconds with a return value of 0.

```

C:\Users\sivas\OneDrive\Documents\ds-delete at end in single link list.exe
Enter the total number of nodes: 5
Enter the data of node 1: 1
Enter the data of node 2: 3
Enter the data of node 3: 5
Enter the data of node 4: 7
Enter the data of node 5: 9
SINGLY LINKED LIST CREATED SUCCESSFULLY

Data in the list
Data = 1
Data = 3
Data = 5
Data = 7
Data = 9

Press 1 to delete last node: 1
SUCCESSFULLY DELETED LAST NODE OF LIST

Data in the list
Data = 1
Data = 3
Data = 5
Data = 7

-----
Process exited after 11.61 seconds with return value 0
Press any key to continue . . .

```

Delete at any position in single link list:

```

#include <stdio.h>

#include <stdlib.h>

struct node {

    int data;

    struct node *next;

} *head;

void createList(int n);

void deleteMiddleNode(int position);

void displayList();

int main()

{

    int n, position;

```

```

printf("Enter the total number of nodes: ");
scanf("%d", &n);
createList(n);
printf("\nData in the list \n");
displayList();
printf("\nEnter the node position you want to delete: ");
scanf("%d", &position);
deleteMiddleNode(position);
printf("\nData in the list \n");
displayList();
return 0;
}

void createList(int n)
{
    struct node *newNode, *temp;
    int data, i;
    head = (struct node *)malloc(sizeof(struct node));
    if(head == NULL)
    {
        printf("Unable to allocate memory.");
    }
    else
    {
        printf("Enter the data of node 1: ");
        scanf("%d", &data);
        head->data = data; // Link the data field with data
        head->next = NULL; // Link the address field to NULL
        temp = head;
        for(i=2; i<=n; i++)
        {
            newNode = (struct node *)malloc(sizeof(struct node));

```

```

        if(newNode == NULL)
        {
            printf("Unable to allocate memory.");
            break;
        }
        else
        {
            printf("Enter the data of node %d: ", i);
            scanf("%d", &data);
            newNode->data = data;
            newNode->next = NULL;
            temp->next = newNode;
            temp = temp->next;
        }
    }
    printf("SINGLY LINKED LIST CREATED SUCCESSFULLY\n");
}

void deleteMiddleNode(int position)
{
    int i;
    struct node *toDelete, *prevNode;
    if(head == NULL)
    {
        printf("List is already empty.");
    }
    else
    {
        toDelete = head;
        prevNode = head;
        for(i=2; i<=position; i++)

```



```

{
    prevNode = toDelete;
    toDelete = toDelete->next;
    if(toDelete == NULL)
        break;
}
if(toDelete != NULL)
{
    if(toDelete == head)
        head = head->next;
    prevNode->next = toDelete->next;
    toDelete->next = NULL;
    free(toDelete);
    printf("SUCCESSFULLY DELETED NODE FROM MIDDLE OF LIST\n");
}
else
{
    printf("Invalid position unable to delete.");
}
}

void displayList()
{
    struct node *temp;
    if(head == NULL)
    {
        printf("List is empty.");
    }
    else
    {
        temp = head;

```

```

while(temp != NULL)
{
    printf("Data = %d\n", temp->data);
    temp = temp->next;
}
}
}

```

The screenshot shows a Windows command prompt window titled "C:\Users\sivas\OneDrive\Documents\ds-delete at any position in single link list.exe". The program prompts the user to enter the total number of nodes (5) and the data for each node (9, 8, 7, 6, 5). It then displays the data in the list. The user is prompted to enter the node position to delete (3), and the program successfully deletes the node from the middle of the list. The final data in the list is 9, 8, 6, 5. The program exits after 10.68 seconds with a return value of 0.

```

C:\Users\sivas\OneDrive\Documents\ds-delete at any position in single link list.exe
Enter the total number of nodes: 5
Enter the data of node 1: 9
Enter the data of node 2: 8
Enter the data of node 3: 7
Enter the data of node 4: 6
Enter the data of node 5: 5
SINGLY LINKED LIST CREATED SUCCESSFULLY

Data in the list
Data = 9
Data = 8
Data = 7
Data = 6
Data = 5

Enter the node position you want to delete: 3
SUCCESSFULLY DELETED NODE FROM MIDDLE OF LIST

Data in the list
Data = 9
Data = 8
Data = 6
Data = 5

-----
Process exited after 10.68 seconds with return value 0
Press any key to continue . . .

```

Searching element in single link list:

```

#include <stdio.h>

#include <stdlib.h>

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

void create(struct node **);
int search(struct node *, int);
void release(struct node **);

```

```

void display(struct node *);

int main()
{
    struct node *p = NULL;
    int key, result;
    printf("Enter data into the list\n");
    create(&p);
    printf("Displaying the nodes in the list:\n");
    display(p);
    printf("Enter key to search in the list: ");
    scanf("%d", &key);
    result = search(p, key);
    if (result)
    {
        printf("%d found in the list.\n", key);
    }
    else
    {
        printf("%d not found in the list.\n", key);
    }
    release(&p);
    return 0;
}

int search(struct node *head, int key)
{
    while (head != NULL)
    {
        if (head->num == key)
        {
            return 1;
        }
    }
}

```

```

        head = head->next;
    }
    return 0;
}

void create(struct node **head)
{
    int c, ch;
    struct node *temp, *rear;
    do
    {
        printf("Enter number: ");
        scanf("%d", &c);
        temp = (struct node *)malloc(sizeof(struct node));
        temp->num = c;
        temp->next = NULL;
        if (*head == NULL)
        {
            *head = temp;
        }
        else
        {
            rear->next = temp;
        }
        rear = temp;
        printf("Do you wish to continue [1/0]: ");
        scanf("%d", &ch);
    } while (ch != 0);
    printf("\n");
}

```

```

void display(struct node *p)

```

```

{
    while (p != NULL)
    {
        printf("%d\t", p->num);

        p = p->next;
    }

    printf("\n");
}

void release(struct node **head)
{
    struct node *temp = *head;

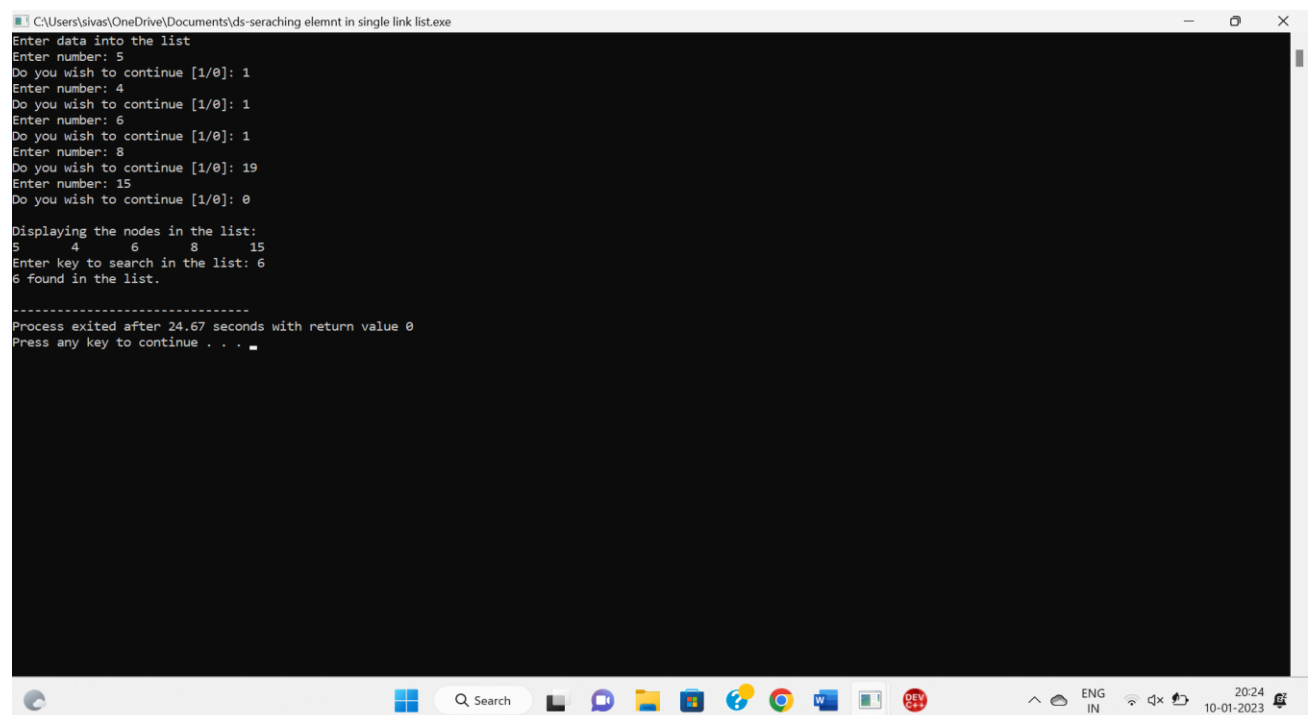
    *head = (*head)->next;

    while ((*head) != NULL)
    {
        free(temp);

        temp = *head;

        (*head) = (*head)->next;
    }
}

```



```

C:\Users\sivas\OneDrive\Documents\ds-seraching elemnt in single link list.exe
Enter data into the list
Enter number: 5
Do you wish to continue [1/0]: 1
Enter number: 4
Do you wish to continue [1/0]: 1
Enter number: 6
Do you wish to continue [1/0]: 1
Enter number: 8
Do you wish to continue [1/0]: 19
Enter number: 15
Do you wish to continue [1/0]: 0

Displaying the nodes in the list:
5 4 6 8 15
Enter key to search in the list: 6
6 found in the list.

-----
Process exited after 24.67 seconds with return value 0
Press any key to continue . . .

```

Implement stack operations :

```
#include<stdio.h>

int stack[100],choice,n,top,x,i;

void push(void);

void pop(void);

void display(void);

int main()
{
    top=-1;

    printf("\n Enter the size of STACK[MAX=100]:");

    scanf("%d",&n);

    printf("\n\t STACK OPERATIONS USING ARRAY");

    printf("\n\t-----");

    printf("\n\t 1.PUSH\n\t 2.POP\n\t 3.DISPLAY\n\t 4.EXIT");

    do
    {
        printf("\n Enter the Choice:");

        scanf("%d",&choice);

        switch(choice)
        {
            case 1:
            {
                push();

                break;
            }

            case 2:
            {
                pop();

                break;
            }
        }
    }
```

```

        case 3:
        {
            display();
            break;
        }
        case 4:
        {
            printf("\n\t EXIT POINT ");
            break;
        }
        default:
        {
            printf ("\n\t Please Enter a Valid Choice(1/2/3/4)");
        }
    }
}
while(choice!=4);
return 0;
}

void push()
{
    if(top>=n-1)
    {
        printf("\n\tSTACK is over flow");

    }
    else
    {
        printf(" Enter a value to be pushed:");
        scanf("%d",&x);
        top++;
    }
}

```

```

        stack[top]=x;
    }
}
void pop()
{
    if(top<=-1)
    {
        printf("\n\t Stack is under flow");
    }
    else
    {
        printf("\n\t The popped elements is %d",stack[top]);
        top--;
    }
}
void display()
{
    if(top>=0)
    {
        printf("\n The elements in STACK \n");
        for(i=top; i>=0; i--)
            printf("\n%d",stack[i]);
        printf("\n Press Next Choice");
    }
    else
    {
        printf("\n The STACK is empty");
    }
}

```



```
C:\Users\sivas\OneDrive\Documents\ds- stack operations push.pop,peak.exe
Enter the size of STACK[MAX=100]:5

STACK OPERATIONS USING ARRAY
-----
1.PUSH
2.POP
3.DISPLAY
4.EXIT
Enter the Choice:1
Enter a value to be pushed:4

Enter the Choice:1
Enter a value to be pushed:7

Enter the Choice:1
Enter a value to be pushed:4

Enter the Choice:1
Enter a value to be pushed:8

Enter the Choice:1
Enter a value to be pushed:9

Enter the Choice:3

The elements in STACK
9
8
4
7
4
Press Next Choice
Enter the Choice:2

The popped elements is 9
Enter the Choice:2

The popped elements is 8
Enter the Choice:2
```

Conversion of infix to postfix expression:

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

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

```
char stack[100];
```

```
int top = -1;
```

```
void push(char x)
```

```
{
```

```
    stack[++top] = x;
```

```
}
```

```
char pop()
```

```
{
```

```
    if(top == -1)
```

```
        return -1;
```

```
    else
```

```
        return stack[top--];
```

```
}
```

```

int priority(char x)
{
    if(x == '(')
        return 0;
    if(x == '+' || x == '-')
        return 1;
    if(x == '*' || x == '/')
        return 2;
    return 0;
}

int main()
{
    char exp[100];
    char *e, x;
    printf("Enter the expression : ");
    scanf("%s",exp);
    printf("\n");
    e = exp;

    while(*e != '\0')
    {
        if(isalnum(*e))
            printf("%c ",*e);
        else if(*e == '(')
            push(*e);
        else if(*e == ')')
        {
            while((x = pop()) != '(')
                printf("%c ", x);
        }
        else

```

```

{
    while(priority(stack[top]) >= priority(*e))
        printf("%c ",pop());
    push(*e);
}
e++;
}
while(top != -1)
{
    printf("%c ",pop());
}return 0;
}

```

The screenshot shows a Windows command prompt window titled "C:\Users\sivas\OneDrive\Documents\ds-conversion of infix to postfix in stack.exe". The user has entered the expression "a*b+c". The program has converted it to the postfix expression "a b * b c * +". The window also displays the message "Process exited after 11.7 seconds with return value 0" and "Press any key to continue . . .". The Windows taskbar at the bottom shows the time as 20:39 on 10-01-2023.

```

C:\Users\sivas\OneDrive\Documents\ds-conversion of infix to postfix in stack.exe
Enter the expression : a*b+c
a b * b c * +
-----
Process exited after 11.7 seconds with return value 0
Press any key to continue . . .

```

Evaluation of arithmetic operators:

```

#include<stdio.h>

#include<conio.h>

int top = -1, stack [100];

main ( ){

    char a[50], ch;

    int i,op1,op2,res,x;

```

```

void push (int);

int pop( );

int eval (char, int, int);

printf("enter a postfix expression:");

gets (a);

for(i=0; a[i]!='\0'; i++){

    ch = a[i];

    if (ch>='0' && ch<='9')

        push('0');

    else{

        op2 = pop ( );

        op1 = pop ( );

        res = eval (ch, op1, op2);

        push (res);

    }

}

x = pop ( );

printf("evaluated value = %d", x);

getch ( );

}

void push (int n){

    top++;

    stack [top] = n;

}

int pop ( ){

    int res ;

    res = stack [top];

    top--;

    return res;

}

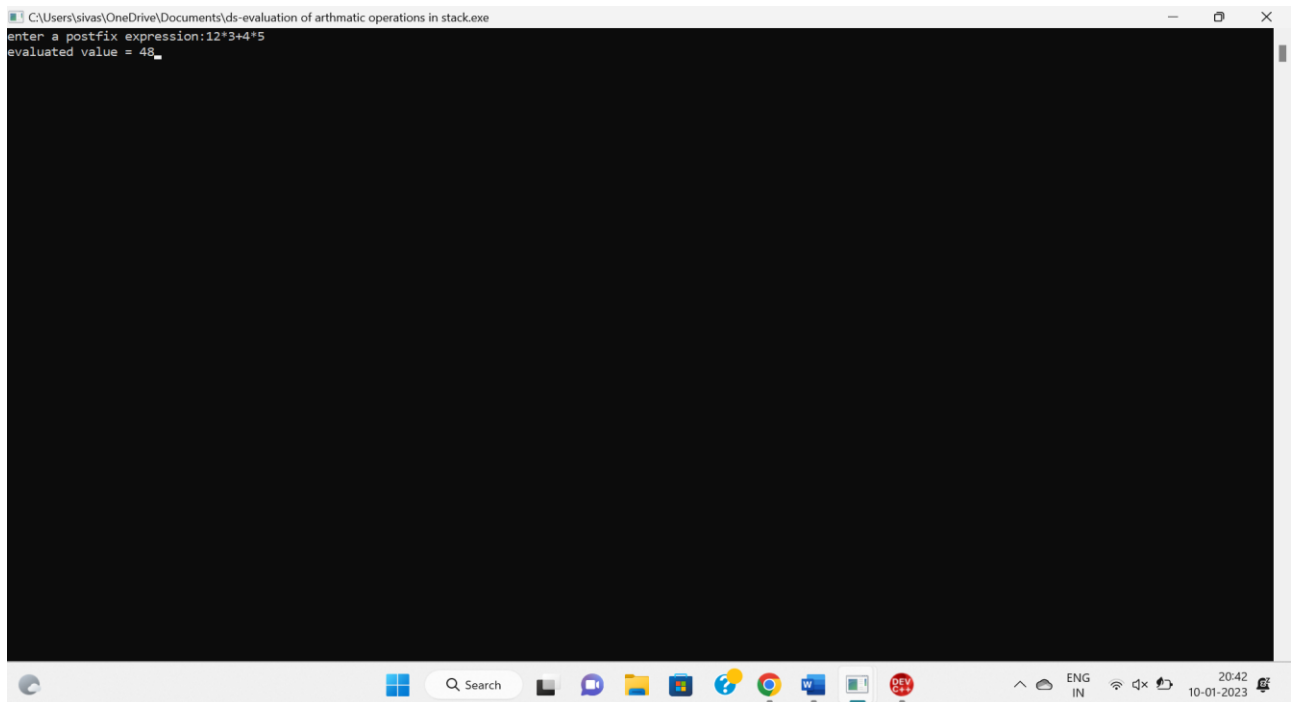
int eval (char ch, int op1, int op2){

```

```

switch (ch){
    case '+' : return (op1+op2);
    case '-' : return (op1-op2);
    case '*' : return (op1*op2);
    case '/' : return (op1/op2);
}
return 0;
}

```



Queue -enqueue,dequeue in an array:

```

#include <stdio.h>

#include<conio.h>

#include<stdlib.h>

#define MAX 50

void insert();

void Delete();

void display();

int queue_array[MAX];

```

```
int rear = - 1;
int front = - 1;
main()
{
    int choice;
    while (1)
    {
        printf("1.Insert element to queue \n");
        printf("2.Delete element from queue \n");
        printf("3.Display all elements of queue \n");
        printf("4.Quit \n");
        printf("Enter your choice : ");
        scanf("%d", &choice);
        switch (choice)
        {
            case 1:
                insert();
                break;
            case 2:
                Delete();
                break;
            case 3:
                display();
                break;
            case 4:
                exit(1);
            default:
                printf("Wrong choice \n");
        }
    }
}
```

```

void insert()
{
    int add_item;
    if (rear == MAX - 1)
        printf("Queue Overflow \n");
    else
    {
        if (front == - 1)
            /*If queue is initially empty */
            front = 0;

        printf("Inset the element in queue : ");
        scanf("%d", &add_item);
        rear = rear + 1;
        queue_array[rear] = add_item;
    }
}

void Delete()
{
    if (front == - 1 || front > rear)
    {
        printf("Queue Underflow \n");
        return ;
    }
    else
    {
        printf("Element deleted from queue is : %d\n", queue_array[front]);
        front = front + 1;
    }
}

void display()

```

```

{
    int i;

    if (front == - 1)

        printf("Queue is empty \n");

    else

    {

        printf("Queue is : \n");

        for (i = front; i <= rear; i++)

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

        printf("\n");

    }

}

```

```

C:\Users\sivas\OneDrive\Documents\ds-queue enqueue,dequeue in array.exe
1.Insert element to queue
2.Delete element from queue
3.Display all elements of queue
4.Quit
Enter your choice : 1
Inset the element in queue : 1
1.Insert element to queue
2.Delete element from queue
3.Display all elements of queue
4.Quit
Enter your choice : 1
Inset the element in queue : 2
1.Insert element to queue
2.Delete element from queue
3.Display all elements of queue
4.Quit
Enter your choice : 1
Inset the element in queue : 3
1.Insert element to queue
2.Delete element from queue
3.Display all elements of queue
4.Quit
Enter your choice : 1
Inset the element in queue : 4
1.Insert element to queue
2.Delete element from queue
3.Display all elements of queue
4.Quit
Enter your choice : 1
Inset the element in queue : 5
1.Insert element to queue
2.Delete element from queue
3.Display all elements of queue
4.Quit
Enter your choice : 3
Queue is :
1 2 3 4 5
1.Insert element to queue
2.Delete element from queue
3.Display all elements of queue

```

Queue-enqueue,dequeue in link list:

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

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

```
struct node
```

```
{
```

```
    int info;
```



```

    struct node *ptr;
}*front,*rear,*temp,*front1;

int frontelement();

void enq(int data);

void deq();

void empty();

void display();

void create();

void queuesize();

int count = 0;

int main()
{
    int no,ch,e;

    printf("\n 1 - Enque");
    printf("\n 2 - Deque");
    printf("\n 3 - Front element");
    printf("\n 4 - Empty");
    printf("\n 5 - Exit");
    printf("\n 6 - Display");
    printf("\n 7 - Queue size");

    create();

    while (1)
    {
        printf("\n Enter choice : ");
        scanf("%d", &ch);

        switch (ch)
        {
            case 1:
                printf("Enter data : ");
                scanf("%d", &no);
                enq(no);

```

```

        break;
case 2:
    deq();
    break;
case 3:
    e = frontelement();
    if (e != 0)
        printf("Front element : %d", e);
    else
        printf("\n No front element in Queue as queue is empty");
    break;
case 4:
    empty();
    break;
case 5:
    exit(0);
case 6:
    display();
    break;
case 7:
    queuesize();
    break;
default:
    printf("Wrong choice, Please enter correct choice ");
    break;
}
}
}
void create()
{
    front = rear = NULL;

```

```

}

void queuesize()
{
    printf("\n Queue size : %d", count);
}

void enq(int data)
{
    if (rear == NULL)
    {
        rear = (struct node *)malloc(1*sizeof(struct node));
        rear->ptr = NULL;
        rear->info = data;
        front = rear;
    }
    else
    {
        temp=(struct node *)malloc(1*sizeof(struct node));
        rear->ptr = temp;
        temp->info = data;
        temp->ptr = NULL;

        rear = temp;
    }
    count++;
}

void display()
{
    front1 = front;
    if ((front1 == NULL) && (rear == NULL))
    {
        printf("Queue is empty");
    }
}

```

```

        return;
    }
    while (front1 != rear)
    {
        printf("%d ", front1->info);
        front1 = front1->ptr;
    }
    if (front1 == rear)
        printf("%d", front1->info);
}

void deq()
{
    front1 = front;
    if (front1 == NULL)
    {
        printf("\n Error: Trying to display elements from empty queue");
        return;
    }
    else
        if (front1->ptr != NULL)
        {
            front1 = front1->ptr;
            printf("\n Dequed value : %d", front->info);
            free(front);
            front = front1;
        }
    else
    {
        printf("\n Dequed value : %d", front->info);
        free(front);
        front = NULL;
    }
}

```

```

        rear = NULL;
    }

    count--;
}

int frontelement()
{
    if ((front != NULL) && (rear != NULL))
        return(front->info);
    else
        return 0;
}

void empty()
{
    if ((front == NULL) && (rear == NULL))
        printf("\n Queue empty");
    else
        printf("Queue not empty");
}

```

```

C:\Users\sivas\OneDrive\Documents\ds-queue enqueue,dequeue in link list.exe
Enter choice : 1
Enter data : 3

Enter choice : 1
Enter data : 4

Enter choice : 1
Enter data : 5

Enter choice : 6
1 2 3 4 5
Enter choice : 2

Dequed value : 1
Enter choice : 2

Dequed value : 2
Enter choice : 2

Dequed value : 3
Enter choice : 7

Queue size : 2
Enter choice : 4
Queue not empty
Enter choice : 5

-----
Process exited after 34.2 seconds with return value 0
Press any key to continue . . .

```