**INDEX**

|  |  |  |
| --- | --- | --- |
| **S No.** | **Programs** | **Signature** |
| 1. | Write a program for traversing an array. |  |
| 2. | Write a program for linear search. |  |
| 3. | Write a program for binary search. |  |
| 4. | Write a program for bubble sorting. |  |
| 5. | Write a program for insertion sorting. |  |
| 6. | Write a program for selection sorting. |  |
| 7. | Write a program for merge sorting. |  |
| 8. | Write a program for inserting an element in an array. |  |
| 9. | Write a program for deleting an element in an array. |  |
| 10. | Write a program to print a sphere matrix. |  |
| 11. | Write a program for linked list. |  |
| 12. | Write a program for searching in singly linked list**.** |  |
| 13. | Write a program to insert and search in a link list. |  |
| 14. | Write a programfor delete an item in link list. |  |
| 15. | Write a programfor delete a node in link list. |  |
| 16. | Write a program to divide a link list into two parts. |  |
| 17. | Write a program for insertion in circular link list. |  |
| 18. | Write a program for searching in circular link list. |  |
| 19. | Write a program for insertion element at beginning random and end. |  |
| 20. | Write a program for reversing a link list. |  |
| 21. | Write a program to search a node in doubly link list. |  |
| 22. | Write a program for sorted link list. |  |
| 23. | Write a program to perform stack function using push, pop, peek. |  |
| 24. | Write a program for stack postfix and infix. |  |
| 25. | Write a program to insert and delete an item from Queue. |  |
| 26. | Write a program to implement circular Queue. |  |
| 27. | Write a program to implement Queue using link list. |  |
|  |  |  |

Program 1:- Write a program for traversing an array.

#include<stdio.h>

void travers\_arr(int arr[],int size){

    for (int i = 0; i < size; i++)

    {

        printf("%d ",arr[i]);

    }

    printf("\n");

}

int main(){

    int arr[100],i,n;

    printf("Enter size of Array: ");

    scanf("%d",&n);

    printf("Enter element of Array: ");

    for ( i = 0; i < n; i++)

    {

       scanf("%d",&arr[i]);

    }

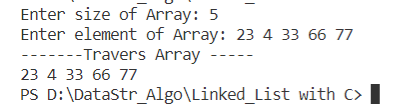
    printf("-------Travers Array -----\n");

    travers\_arr(arr,n);

    return 0;

}

Output:-



Program 2:- Write a program for linear search.

#include<stdio.h>

int linearSearch(int arr[],int size,int element){

  for (int i = 0; i < size; i++)

  {

    if (arr[i]==element)

    {

      return i;

    }

  }

  return -1;

}

int main(){

  int arr[100],element,n;

  printf("Enter Size of Array: ");

  scanf("%d",&n);

  printf("Enter Array Elements: ");

  for (int i = 0; i <n; i++)

  {

    scanf("%d",&arr[i]);

  }

  int size=sizeof(arr)/sizeof(int);

  printf("Enter Element you want to Search in Array ");

  scanf("%d",&element);

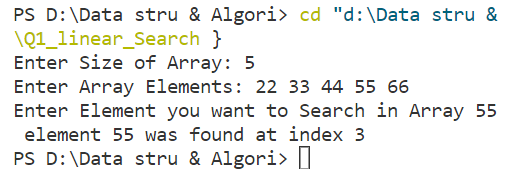
  int SearchIndex = linearSearch(arr,size,element);

  printf(" element %d was found at index %d \n",element,SearchIndex);

  return 0;

}

Output:-



Program 3:- Write a program for binary search.

#include<stdio.h>

int binarySearch(int arr[],int size,int element){

     int beg,mid,end;

     beg=0;

    end=size-1;

    while (beg<=end){

        mid = (beg+end)/2;

        if (arr[mid]==element)

        {

            return mid;

        }

        else if(arr[mid]<element){

            beg = mid+1;

        }else{

            end = mid -1;

        }

    }

    return -1;

}

int main(){

    int arr[100],n,element;

    printf("Enter Size of Array: ");

    scanf("%d",&n);

    printf("Enter Array Elements ");

    for (int i = 0; i <n; i++){

        scanf("%d ",&arr[i]);

    }

    scanf("%d",&element);

    int searchIndex;

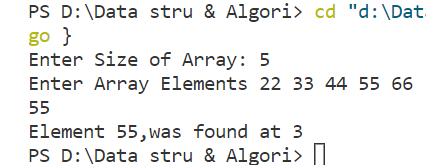
    searchIndex = binarySearch(arr,n,element);

    printf("Element %d,was found at %d",element,searchIndex);

    return 0;

}

Output:-



Program 4:- Write a program for bubble sorting.

#include <stdio.h>

void Display(int arr[], int Size)

{

    for (int i = 0; i < Size; i++)

    {

        printf("%d ", arr[i]);

    }

    printf("\n");

}

int main()

{

    int arr[100],n;

    printf("Enter Size of Array: ");

    scanf("%d",&n);

    printf("Enter element of array: ");

    for (int i = 0; i <n; i++)

    {

        scanf("%d",&arr[i]);

    }

    Display(arr, n);

    // bubble Sort;

    int temp;

    for (int i = 0; i < n - 1; i++) // no of pass

    // if we have 6 element of Array

    // then no of pass(it mean's layer of steps) is 5

    // so we use n-1; where n is size of Array

    {

        for (int j = 0; j < n - 1 - i; j++) // for no of comparison

        // it mean 1 ke liye n-1 comparison

        // it mean 2 ke liye n-2 comparison

        // it mean 3 ke liye n-3 comparison

        // it for n ke liye n-1-i comparison  where i=0;

        {

            if (arr[j] > arr[j + 1])

            {

                int temp;

                temp = arr[j];       // arr[j] first element of array

                arr[j] = arr[j + 1]; // arr[j+1] next element of Array

                arr[j + 1] = temp;

            }

        }

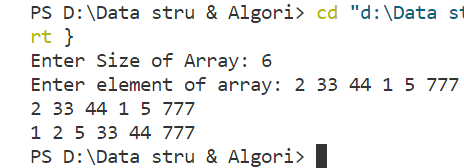
    }

    Display(arr, n);

    return 0;

}

Output:-



Program5:- Write a program for insertion sorting.

#include<stdio.h>

void Display(int arr[],int size){

    for (int i = 0; i < size; i++)

    {

        printf("%d ",arr[i]);

    }

    printf("\n");

}

int main(){

    int arr[100],i,n;

    printf("Enter size of Array: ");

    scanf("%d",&n);

    printf("Enter element of Array: ");

    for ( i = 0; i < n; i++)

    {

       scanf("%d",&arr[i]);

    }

    int temp,j;

    for ( i = 0; i <=n-1; i++)

    {

        for ( j = 0; j <n-i-1 ; j++)

        {

            if (arr[j]>arr[j+1])

            {

                temp = arr[j];

                arr[j] = arr[j+1];

                arr[j+1] =temp;

            }

        }

    }

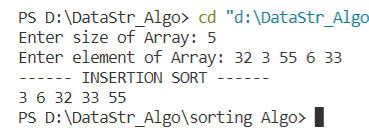
    printf("------ INSERTION SORT ------ \n");

    Display(arr,n);

    return 0;

}

Output:-



Program 6:- Write a program for selection sorting.

#include<stdio.h>

void display(int arr[], int size){

    for (int i = 0; i<size; i++)

    {

        printf("%d ",arr[i]);

    }

    printf("\n");

}

int main(){

    int A[100],n,i,j;

    int indexOfMin,temp;

    printf("-----Selectoin Sort------- \n");

    printf("Enter size of Array you want to sort: ");

    scanf("%d",&n);

    printf("Enter array element : ");

    for (int i = 0; i <n; i++)

    {

        scanf("%d",&A[i]);

    }

    for ( i = 0; i < n-1; i++)

    {

        indexOfMin = i;

        for ( j = i+1; j <=n-1; j++)

        {

            if (A[j]<A[indexOfMin])

            {

                indexOfMin = j;

            }

        }

        // swap A[]

        temp =A[i];

        A[i] = A[indexOfMin];

        A[indexOfMin] = temp;

    }

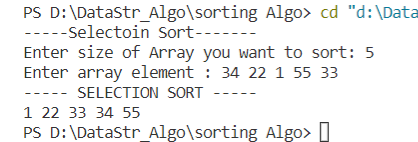
    printf("----- SELECTION SORT -----\n");

    display(A,n);

    return 0;

}

Output:-



Program7:- Write a program for merge sorting.

#include <stdio.h>

void Display(int arr[], int size)

{

    for (int i = 0; i < size; i++)

    {

        printf("%d ", arr[i]);

    }

    printf("\n");

}

void merge(int A[], int mid, int low, int high)

{

    int i, j, k;

    int B[100];

    i = low;

    j = mid + 1;

    k = low;

    while (i <= mid && j <= high)

    {

        if (A[i] < A[j])

        {

            B[k] = A[i];

            i++;

            k++;

        }

        else

        {

            B[k] = A[j];

            j++;

            k++;

        }

    }

    while (i <= mid)

    {

        B[k] = A[i];

        k++;

        i++;

    }

    while (j <= high)

    {

        B[k] = A[j];

        k++;

        j++;

    }

    for (int i = low; i <= high; i++)

    {

        A[i] = B[i];

    }

}

void mergeSort(int A[], int low, int high)

{

    int mid;

    // 0 1 2 3

    if (low < high)

    {

        mid = (low + high) / 2;

        mergeSort(A, low, mid);

        mergeSort(A, mid + 1, high);

        merge(A, mid, low, high);

    }

}

int main()

{

    int A[] = {9, 14, 4, 8, 7, 5, 6}, n = 7;

    Display(A, n);

    printf("(--------Merge Sort--------)\n");

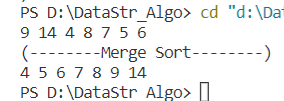
    mergeSort(A, 0, 6); // Assume index start with 0

    Display(A, n);

    return 0;

}

Output:-



Program 8:- Write a program for inserting an element in an array.

#include<stdio.h>

void display(int arr[], int size){

    for (int i = 0; i<size; i++)

    {

        printf("%d ",arr[i]);

    }

    printf("\n");

}

int Insertion(int arr[],int size,int position,int element){

    for (int i = size-1; i>=position-1; i--)

    {

        arr[i+1] = arr[i];

    }

    arr[position-1]=element;

}

int main(){      //n = size of Array

    int arr[100],n,i,j,pos,element; //position is nothing

    printf("Enter Array Size: ");   // index = pos -1; becouse your user don't no Array index

    scanf("%d",&n);

    printf("Enter Array element: ");

    for (int i = 0; i <n; i++){

        scanf("%d",&arr[i]);

    }

    printf("Enter position of Array you want to insert element: ");

    scanf("%d",&pos);

    printf("Enter element of Array you want to insert: ");

    scanf("%d",&element);

    Insertion(arr,n,pos,element);

    n++;

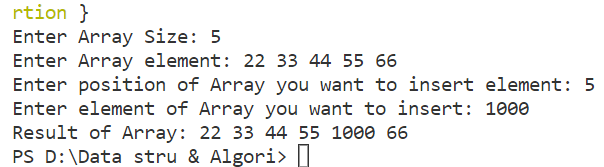
    printf("Result of Array: ");

    display(arr,n);

    return 0;

}

Output:-



Program 9:- Write a program for deleting an element in an array.

#include<stdio.h>

void  DISPLAY( int arr[],int n){

    for (int i = 0; i < n; i++){

        printf("%d ",arr[i]);

    }

    printf("\n");

}

int Deletion\_ele(int arr[],int pos,int size){

    for (int i = pos-1; i <=size-1 ; i++){

        arr[i] = arr[i+1];

    }

}

int main(){

    int arr[100],n,pos;

    printf("Enter Size of Array: ");

    scanf("%d",&n);

    printf("Enter Array element: ");

    for (int i = 0; i <n; i++){

        scanf("%d",&arr[i]);

     }

    printf("Enter Position of Array you want to delete: ");

    scanf("%d",&pos);

    Deletion\_ele(arr,pos,n);

    n--;

    DISPLAY( arr, n);

    return 0;

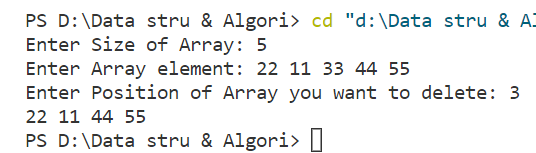
}

//     //size -1 (n-1) is nothing

//     //but when we start array

//     // with 0 so we take n-1

Output:-



Program 10:- Write a program to print a sphere matrix.

#include <stdio.h>

**int** main()

{

    // Sparse matrix having size 4\*5

**int** sparse\_matrix[4][5] =

    {

        {0 , 0 , 6 , 0 , 9 },

        {0 , 0 , 4 , 6 , 0 },

        {0 , 0 , 0 , 0 , 0 },

        {0 , 1 , 2 , 0 , 0 }

    };

   // size of matrix

**int** size = 0;

**for**(**int** i=0; i<4; i++)

    {

**for**(**int** j=0; j<5; j++)

        {

**if**(sparse\_matrix[i][j]!=0)

            {

                size++;

            }

        }

    }

   // Defining final matrix

**int** matrix[3][size];

**int** k=0;

   // Computing final matrix

**for**(**int** i=0; i<4; i++)

    {

**for**(**int** j=0; j<5; j++)

        {

**if**(sparse\_matrix[i][j]!=0)

            {

                matrix[0][k] = i;

                matrix[1][k] = j;

                matrix[2][k] = sparse\_matrix[i][j];

                k++;

            }

      }

    }

   // Displaying the final matrix

**for**(**int** i=0 ;i<3; i++)

    {

**for**(**int** j=0; j<size; j++)

        {

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

            printf("\t");

        }

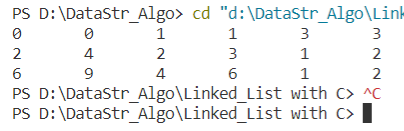
        printf("\n");

    }

**return** 0;

}

Output:-



Program 11:- Write a program for linked list.

#include<stdio.h>

#include<stdlib.h>

struct node

{

    int num;

    struct node \*nextptr;

}\*stnode;

void createNodeList(int n);

void display();

int main(){

    int n;

    printf("\n\n Linked List : To create and display Singly Linked List :\n");

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

    printf("Enter a number of nodes: ");

    scanf("%d",&n);

    createNodeList(n);

    printf("\n Data entered in the list:  \n");

    display();

    return 0;

}

void createNodeList(int n){

    struct node \*fnNode,\*temp;

    int num,i;

    stnode =(struct node\*)malloc(sizeof(struct node));

    if (stnode==NULL)

    {

        printf("memory not allocted ");

    }else{

        printf("Input data for node1 : ");

        scanf("%d",&num);

        stnode->num = num;

        stnode->nextptr = NULL;

        temp = stnode;

        // Creating n nodes and adding to linked list

        for ( i = 2; i <=n; i++)

        {

            fnNode = (struct node\*)malloc(sizeof(struct node));

            if (fnNode==NULL)

            {

                printf("memory can not be allocated ");

                break;

            }

            else{

                printf("Input data for %d : ",i);

                scanf("%d",&num);

                fnNode ->num =num;

                fnNode->nextptr =NULL;

                temp ->nextptr = fnNode;

                temp = temp->nextptr;

            }

        }

    }

}

void display(){

    struct node \*temp;

    if (stnode==NULL)

    {

        printf(" List is empty ");

    }

    else{

        temp = stnode;

        while (temp!=NULL)

        {

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

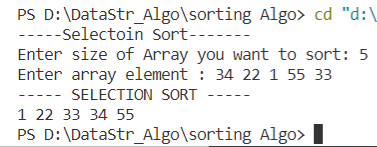
            temp = temp->nextptr;

        }

    }

}

Output:-



Program 12:- Write a program for searching in singly linked list**.**

#include <stdio.h>

#include <stdlib.h>

void create(int);

void search();

struct node

{

    int data;

    struct node \*next;

};

struct node \*head;

void main()

{

    int choice, item, loc;

    do

    {

        printf("\n1.Create\n2.Search\n3.Exit\n4.Enter your choice?");

        scanf("%d", &choice);

        switch (choice)

        {

        case 1:

            printf("\nEnter the item\n");

            scanf("%d", &item);

            create(item);

            break;

        case 2:

            search();

        case 3:

            exit(0);

            break;

        default:

            printf("\nPlease enter valid choice\n");

        }

    } while (choice != 3);

}

void create(int item)

{

    struct node \*ptr = (struct node \*)malloc(sizeof(struct node \*));

    if (ptr == NULL)

    {

        printf("\nOVERFLOW\n");

    }

    else

    {

        ptr->data = item;

        ptr->next = head;

        head = ptr;

        printf("\nNode inserted\n");

    }

}

void search()

{

    struct node \*ptr;

    int item, i = 0, flag;

    ptr = head;

    if (ptr == NULL)

    {

        printf("\nEmpty List\n");

    }

    else

    {

        printf("\nEnter item which you want to search?\n");

        scanf("%d", &item);

        while (ptr != NULL)

        {

            if (ptr->data == item)

            {

                printf("item found at location %d ", i + 1);

                flag = 0;

            }

            else

            {

                flag = 1;

            }

            i++;

            ptr = ptr->next;

        }

        if (flag == 1)

        {

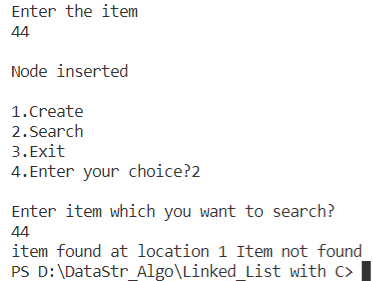
            printf("Item not found\n");

        }

    }

}

Output:-



Program 13:- Write a program to insert and search in a link list.

#include<stdio.h>

#include<stdlib.h>

struct node

{

    int data;

    struct node\*next;

};

void display(struct node\*ptr){

    while (ptr!=NULL)

    {

        printf("Element of list is %d \n",ptr->data);

        ptr = ptr ->next;

    }

}

struct node \* insertAtBeginning(struct node \*head,int item){

    struct node \*ptr = (struct node\*)malloc(sizeof(struct node));

    ptr->data =item;

    ptr->next = head;

    return ptr;

}

int main(){

    struct node \*head;

    struct node \*second;

    struct node \*third;

    head = (struct node\*)malloc(sizeof(struct node));

    second = (struct node\*)malloc(sizeof(struct node));

    third = (struct node\*)malloc(sizeof(struct node));

    head ->data =1000;

    head ->next = second;

    second ->data =1300;

    second ->next = third;

    third ->data =1040;

    third ->next = NULL;

    display(head);

    printf("-----------insert node At Beginning-----------");

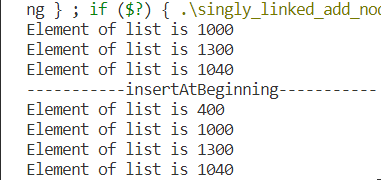
    head = insertAtBeginning(head,400);

    display(head);

    return 0;

}

Output:-



Program 15:- Write a programfor delete a node in link list.

#include <stdio.h>

#include <stdlib.h>

void create(int);

void last\_delete();

struct node

{

    int data;

    struct node \*next;

    struct node \*prev;

};

struct node \*head;

void main()

{

    int choice, item;

    do

    {

        printf("1.Append List\n2.Delete node from end\n3.Exit\n4.Enter your choice?");

        scanf("%d", &choice);

        switch (choice)

        {

        case 1:

            printf("\nEnter the item\n");

            scanf("%d", &item);

            create(item);

            break;

        case 2:

            last\_delete();

            break;

        case 3:

            exit(0);

            break;

        default:

            printf("\nPlease enter valid choice\n");

        }

    } while (choice != 3);

}

void create(int item)

{

    struct node \*ptr = (struct node \*)malloc(sizeof(struct node));

    if (ptr == NULL)

    {

        printf("\nOVERFLOW\n");

    }

    else

    {

        if (head == NULL)

        {

            ptr->next = NULL;

            ptr->prev = NULL;

            ptr->data = item;

            head = ptr;

        }

        else

        {

            ptr->data = item;

            ptr->prev = NULL;

            ptr->next = head;

            head->prev = ptr;

            head = ptr;

        }

        printf("\nNode Inserted\n");

    }

}

void last\_delete()

{

    struct node \*ptr;

    if (head == NULL)

    {

        printf("\n UNDERFLOW\n");

    }

    else if (head->next == NULL)

    {

        head = NULL;

        free(head);

        printf("\nNode Deleted\n");

    }

    else

    {

        ptr = head;

        if (ptr->next != NULL)

        {

            ptr = ptr->next;

        }

        ptr->prev->next = NULL;

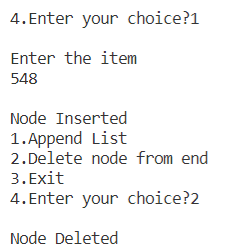
        free(ptr);

        printf("\nNode Deleted\n");

    }

}

Output:-



Program 16:- Write a program to divide a link list into two parts.

#include <stdio.h>

#include <stdlib.h>

struct node {

   int data;

   struct node \*next;

};

struct node \*even = NULL;

struct node \*odd = NULL;

struct node \*list = NULL;

//Create Linked List

void insert(int data) {

   // Allocate memory for new node;

   struct node link = (struct node) malloc(sizeof(struct node));

   struct node \*current;

   link->data = data;

   link->next = NULL;

   if(list == NULL) {

      list = link;

      return;

   }

   current = list;

   while(current->next!=NULL)

      current = current->next;

   // Insert link at the end of the list

   current->next = link;

}

void display(struct node \*head) {

   struct node \*ptr = head;

   printf("[head] =>");

   //start from the beginning

   while(ptr != NULL) {

      printf(" %d =>",ptr->data);

      ptr = ptr->next;

   }

   printf(" [null]\n");

}

void split\_list() {

   // Allocate memory for new node;

   struct node \*link;

   struct node \*current;

   while(list != NULL) {

      struct node link = (struct node) malloc(sizeof(struct node));

      link->data = list->data;

      link->next = NULL;

      if(list->data%2 == 0) {

         if(even == NULL) {

            even = link;

            list = list->next;

            continue;

         } else {

            current = even;

            while(current->next != NULL)

            current = current->next;

            // Insert link at the end of the list

            current->next = link;

         }

         list = list->next;

      } else {

         if(odd == NULL) {

            odd = link;

            list = list->next;

            continue;

         } else {

            current = odd;

            while(current->next!=NULL)

            current = current->next;

            // Insert link at the end of the list

            current->next = link;

         }

         list = list->next;

      }

   }

}

int main() {

   int i;

   for(i = 1; i <= 10; i++)

      insert(i);

   printf("Complete list: \n");

   display(list);

   split\_list();

   printf("\nOdd  : ");

   display(odd);

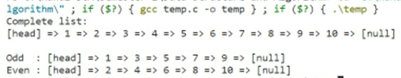
   printf("Even : ");

   display(even);

   return 0;

}

Output:-



Program 17:- Write a program for insertion in circular link list.

#include<stdio.h>

#include<stdlib.h>

void beg\_insert(int);

struct node

{

    int data;

    struct node \*next;

};

struct node \*head;

void main ()

{

    int choice,item;

    do

    {

        printf("\nEnter the item which you want to insert?\n");

        scanf("%d",&item);

        beg\_insert(item);

        printf("\nPress 0 to insert more ?\n");

        scanf("%d",&choice);

    }while(choice == 0);

}

void beg\_insert(int item)

{

    struct node \*ptr = (struct node \*)malloc(sizeof(struct node));

    struct node \*temp;

    if(ptr == NULL)

    {

        printf("\nOVERFLOW");

    }

    else

    {

        ptr -> data = item;

        if(head == NULL)

        {

            head = ptr;

            ptr -> next = head;

        }

        else

        {

            temp = head;

            while(temp->next != head)

                temp = temp->next;

            ptr->next = head;

            temp -> next = ptr;

            head = ptr;

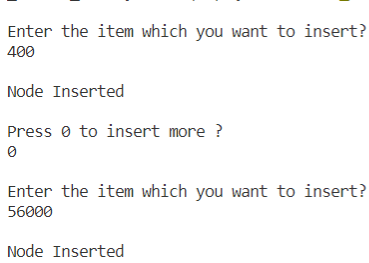
        }

    printf("\nNode Inserted\n");

    }

}

Output:-



Program 18:- Write a program for searching in circular link list.

Code:

#include <stdio.h>

#include <stdlib.h>

struct node {

    int num;

    struct node \* nextptr;

}\*stnode;

void ClListcreation(int n);

void displayClList();

void search();

int main()

{

    int n;

    stnode = NULL;

 printf("\n\n Circular Linked List : Create and display a circular linked list :\n");

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

    printf(" Input the number of nodes : ");

    scanf("%d", &n);

    ClListcreation(n);

    displayClList();

    search();

    return 0;

}

void ClListcreation(int n)

{

    int i, num;

    struct node \*preptr, \*newnode;

    if(n >= 1)

    {

        stnode = (struct node \*)malloc(sizeof(struct node));

        printf(" Input data for node 1 : ");

        scanf("%d", &num);

        stnode->num = num;

        stnode->nextptr = NULL;

        preptr = stnode;

        for(i=2; i<=n; i++)

        {

            newnode = (struct node \*)malloc(sizeof(struct node));

            printf(" Input data for node %d : ", i);

            scanf("%d", &num);

            newnode->num = num;

            newnode->nextptr = NULL;

            preptr->nextptr = newnode;

            preptr = newnode;

        }

        preptr->nextptr = stnode;

    }

}

void displayClList()

{

    struct node \*tmp;

    int n = 1;

    if(stnode == NULL)

    {

        printf(" No data found in the List yet.");

    }

    else

    {

        tmp = stnode;

        printf("\n\n Data entered in the list are :\n");

        do {

            printf(" Data %d = %d\n", n, tmp->num);

            tmp = tmp->nextptr;

            n++;

        }while(tmp != stnode);

    }

}

void search()

{

    int item,i;

    struct node \*temp;

    printf("Enter the element you want to search");

    scanf("%d",&item);

    temp= stnode;

    if(temp ->num == item)

    {

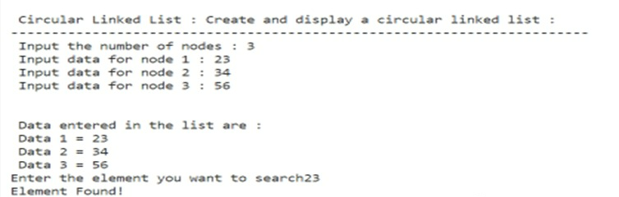
        printf("Element Found!");

    }

    temp= temp-> nextptr;

}

Output:-



Program 19:- Write a program for insertion element at beginning random and end.

#include<stdio.h>

#include<stdlib.h>

struct node

{

    int data;

    struct node\*next;

};

void display(struct node\*ptr){

    while (ptr!=NULL)

    {

        printf("Element of list is %d \n",ptr->data);

        ptr = ptr ->next;

    }

}

struct node \* insertAtBeginning(struct node \*head,int item){

    struct node \*ptr = (struct node\*)malloc(sizeof(struct node));

    ptr->data =item;

    ptr->next = head;

    return ptr;

}

// struct node \* insertAtIndex(struct node \*head,int item,int index){

//     struct node \*ptr = (struct node\*)malloc(sizeof(struct node));

//     struct node \*p = head;

//     int i=0;

//     while (i!=index-1)

//     {

//         p=p->next;

//         i++;

//     }

//     ptr ->data = item;

//     ptr->next = p->next;

//     p->next = ptr;

//     return head;

// }

// struct node \* insertAtEnd(struct node \*head,int item){

//     struct node \*ptr = (struct node\*)malloc(sizeof(struct node));

//     ptr->data = item;

//     struct node \*p = head;

//     while (p->next!=NULL)

//     {

//         p=p->next;

//     }

//     p->next =ptr;

//     ptr ->next =NULL;

//     return head;

// }

int main(){

    struct node \*head;

    struct node \*second;

    struct node \*third;

    head = (struct node\*)malloc(sizeof(struct node));

    second = (struct node\*)malloc(sizeof(struct node));

    third = (struct node\*)malloc(sizeof(struct node));

    head ->data =1000;

    head ->next = second;

    second ->data =1300;

    second ->next = third;

    third ->data =1040;

    third ->next = NULL;

    display(head);

    printf("-----------insert node At Beginning-----------");

    head = insertAtBeginning(head,400);

    // printf("--------------insert At Index------------\n");

    // head = insertAtIndex(head,40,1);

    // printf("-------------insert At End--------------\n");

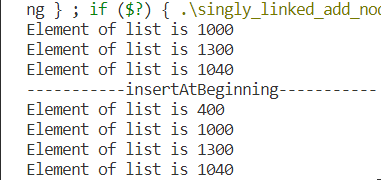
    // head = insertAtEnd(head,400);

    display(head);

    return 0;

}

Output:-



Program 20:- Write a program for reversing a link list.

// Iterative C program to reverse a linked list

#include <stdio.h>

#include <stdlib.h>

/\* Link list node \*/

struct Node {

  int data;

  struct Node\* next;

};

/\* Function to reverse the linked list \*/

static void reverse(struct Node\*\* head\_ref)

{

  struct Node\* prev = NULL;

  struct Node\* current = \*head\_ref;

  struct Node\* next = NULL;

  while (current != NULL) {

    // Store next

    next = current->next;

    // Reverse current node's pointer

    current->next = prev;

    // Move pointers one position ahead.

    prev = current;

    current = next;

  }

  \*head\_ref = prev;

}

/\* Function to push a node \*/

void push(struct Node\*\* head\_ref, int new\_data)

{

  struct Node\* new\_node

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

  new\_node->data = new\_data;

  new\_node->next = (\*head\_ref);

  (\*head\_ref) = new\_node;

}

/\* Function to print linked list \*/

void printList(struct Node\* head)

{

  struct Node\* temp = head;

  while (temp != NULL) {

    printf("%d ", temp->data);

    temp = temp->next;

  }

}

/\* Driver code\*/

int main()

{

  /\* Start with the empty list \*/

  struct Node\* head = NULL;

  push(&head, 20);

  push(&head, 4);

  push(&head, 15);

  push(&head, 85);

  printf("Given linked list\n");

  printList(head);

  reverse(&head);

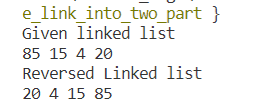
  printf("\nReversed Linked list \n");

  printList(head);

  getchar();

}

Output:-



Program 21:- Write a program to search a node in doubly link list.

#include <stdio.h>

#include <stdlib.h>

void create(int);

void search();

struct node

{

  int data;

  struct node \*next;

  struct node \*prev;

};

struct node \*head;

void main()

{

  int choice, item, loc;

  do

  {

    printf("\n1.Create\n2.Search\n3.Exit\n4.Enter your choice?");

    scanf("%d", &choice);

    switch (choice)

    {

    case 1:

      printf("\nEnter the item\n");

      scanf("%d", &item);

      create(item);

      break;

    case 2:

      search();

    case 3:

      exit(0);

      break;

    default:

      printf("\nPlease enter valid choice\n");

    }

  } while (choice != 3);

}

void create(int item)

{

  struct node \*ptr = (struct node \*)malloc(sizeof(struct node));

  if (ptr == NULL)

  {

    printf("\nOVERFLOW");

  }

  else

  {

    if (head == NULL)

    {

      ptr->next = NULL;

      ptr->prev = NULL;

      ptr->data = item;

      head = ptr;

    }

    else

    {

      ptr->data = item;

      printf("\nPress 0 to insert more ?\n");

      ptr->prev = NULL;

      ptr->next = head;

      head->prev = ptr;

      head = ptr;

    }

    printf("\nNode Inserted\n");

  }

}

void search()

{

  struct node \*ptr;

  int item, i = 0, flag;

  ptr = head;

  if (ptr == NULL)

  {

    printf("\nEmpty List\n");

  }

  else

  {

    printf("\nEnter item which you want to search?\n");

    scanf("%d", &item);

    while (ptr != NULL)

    {

      if (ptr->data == item)

      {

        printf("\nitem found at location %d ", i + 1);

        flag = 0;

        break;

      }

      else

      {

        flag = 1;

      }

      i++;

      ptr = ptr->next;

    }

    if (flag == 1)

    {

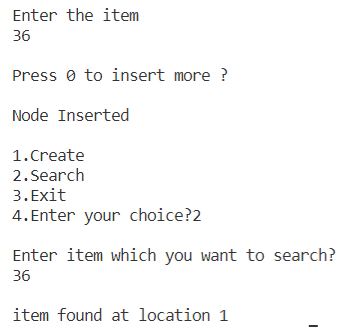
      printf("\nItem not found\n");

    }

  }

}

Output:-



Program 22:- Write a program for sorted link list.

#include <stdio.h>

#include <stdlib.h>

// Create a node

struct Node {

  int data;

  struct Node\* next;

};

// Sort the linked list

void sortLinkedList(struct Node\*\* head\_ref) {

  struct Node \*current = \*head\_ref, \*index = NULL;

  int temp;

  if (head\_ref == NULL) {

  return;

  } else {

  while (current != NULL) {

    // index points to the node next to current

    index = current->next;

    while (index != NULL) {

    if (current->data > index->data) {

      temp = current->data;

      current->data = index->data;

      index->data = temp;

    }

    index = index->next;

    }

    current = current->next;

  }

  }

}

// Print the linked list

void printList(struct Node\* node) {

  while (node != NULL) {

  printf(" %d ", node->data);

  node = node->next;

  }

}

// Driver program

int main() {

  struct Node\* head = NULL;

  insertAtEnd(&head, 1);

  insertAtBeginning(&head, 2);

  insertAtBeginning(&head, 3);

  insertAtEnd(&head, 4);

  insertAfter(head->next, 5);

  printf("Linked list: ");

  printList(head);

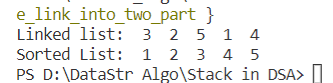
  sortLinkedList(&head);

  printf("\nSorted List: ");

  printList(head);

}

Output:-



Program 23:- Write a program to perform stack function using push, pop, peek.

#include <stdio.h>

int stack[100],i,j,choice=0,n,top=-1;

void push();

void pop();

void show();

void main ()

{

    printf("Enter the number of elements in the stack ");

    scanf("%d",&n);

    printf("\*\*\*\*Stack operations using array\*\*\*\*");

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

    while(choice != 4)

    {

        printf("Chose one from the below options...\n");

        printf("\n1.Push\n2.Pop\n3.Show\n4.Exit");

        printf("\n Enter your choice \n");

        scanf("%d",&choice);

        switch(choice)

        {

            case 1:

            {

                push();

                break;

            }

            case 2:

            {

                pop();

                break;

            }

            case 3:

            {

                show();

                break;

            }

            case 4:

            {

                printf("Exiting....");

                break;

            }

            default:

            {

                printf("Please Enter valid choice ");

            }

        };

    }

}

void push ()

{

    int val;

    if (top == n )

    printf("\n Overflow");

    else

    {

        printf("Enter the value?");

        scanf("%d",&val);

        top = top +1;

        stack[top] = val;

    }

}

void pop ()

{

    if(top == -1)

    printf("Underflow");

    else

    top = top -1;

}

void show()

{

    for (i=top;i>=0;i--)

    {

        printf("%d\n",stack[i]);

    }

    if(top == -1)

    {

        printf("Stack is empty");

    }

}

Output:-

