INDEX

Date .....

S.R.	Date	Programs Name / Units / Exp.	T-Sign.
		Unit-1 Introduction to Data Structure and it's characteristics Array + Unit 2, 3, 4, 5, 6	
1.	07-09-22	Sum of first n Natural No. without Arr.	KR
2.	08-09-22	Sum of first n Natural No. with Arr.	
3.	09-09-22	implement Sorting character Array	
4.	12-09-22	Arrange the Names in Alpha order in 2D	KU
5.	13-09-22	insertion operation at specified Location	
6.	16-09-22	deletion operation at specified Location	
7.	19-09-22	implement creation of Sparse matrix	
8.	20-09-22	Addition of two 2D Arrays	KU
9.	20-09-22	Multiplication of two 2D Arrays	
10.	28-09-22	transpose a 2D Arrays	
11.	29-09-22	implement stack operation using Arr.	
12.	06-10-22	implement Reverse a string Using stack.	
13.		implement stack using Arrays	
14.		implement Stack Using Linked list	
15.		Singly linked list menu driven	
16.		" deletion/insertion at the starting	
17.		" deletion/insertion at end	
18.		" " " at specific position	
19.		" " print the linked list	
20.		" " Maximum among linked list	
21.		" " sorting the linked list	
22.		" " Reverse the linked list	
23.		" " Delete the 1st & last Element	
24.		Insert & Delete a node in DOBLY linked list	
25.		WAP to implement selection sort	
26.		WAP to " insertion sort	
27.		" " " Bubble sort, Merge sort	

Spiral

Date 05-09-22

1. WAP to Print Sum of first n natural numbers without using an Array.

=) /* sum of n Natural Numbers by For Loop */

```
#include<stdio.h>
#include<conio.h>
void main()
{
    int num, i, sum = 0; //declare local variables
    Printf("enter a positive Number :- ");
    Scanf("%d", &num); //take any positive number

//executes until the Condition remains true.

    for(i=0; i<= num; i++)
    {
        SUM= sum + i;
    }
//display the sum of natural number.

    Printf("\n Sum of the first %d number is: %d",
            num, sum);

    getch();
}
```

Spiral

Date .....

Output :-

Enter a Positive Number : 25

Sum of the first 25 Number is : 325

Date 08-09-22

2. WAP to Print sum of first n natural numbers with using an Array.

⇒ /* sum of first n Natural numbers using an array */

```
#include <stdio.h>
int main()
{
    // declare & initialize local variable
    int num, sum=0, i, array[50];
    printf("Enter a positive number as :- ");
    scanf("%d", &num); // take a positive number
    printf("\n Enter the number one by one :\n");
    For (i=0; i<num; i++)
    {
        Scanf ("%d", &array[i]); //read value one by one
        Sum = Sum + array[i]; // store number into sum
    }

    Printf ("Sum of the given number is = %d\n", sum);

    return 0;
}
```

Date .....

Output :-

Enter a positive number :- 5

Enter the number one by one :

2
4
5
6
7

Sum of the given number is = 24

Date 09-09-22

3. WAP to implement sorting character Array.

⇒ /* implement sorting character Array */

```
#include <stdio.h>
#include< conio.h>
int main (void) {
    char string[] = "simplyeasylearning";
    char temp;

    int i, j;
    int n = strlen (string);
    printf ("string before sorting - %s \n", string);

    For (i=0; i<n-1; i++) {
        For (j= i+1; j<n; j++) {
            if (string [i] > string [j]) {
                temp= string [i];
                string [i] = string [j];
                string [j] = temp;
            }
        }
    }
    printf ("string after sorting - %s \n", string);
    return 0;
```

Date .....

Output :-

String before sorting - Simplyeasylearning

String after sorting - aaeegiillmmnprssyy

Date 12-09-22

4. WAP to arrange the names in alphabetical order stored in 2D Array.

⇒ /* Arrang names in Alphabetical order stored in 2D Array */

```
#include<stdio.h>
#include<string.h>
main()
{
    int i, j, n;
    char str[100][100], s[100];
    printf("Enter number of names:\n");
    scanf("%d", &n);
    printf("Enter names in any order:\n");
    For(i=0; i<n; i++){
      For(j=i+1; j<n; j++){
          if(strcmp(str[i], str[j])>0){
              Strcpy(s, str[i]);
              Strcpy(str[i], str[j]);
              Strcpy(str[j], s);
          }
      }
    }
}
printf("\n The sorted order of names are:\n");
For(i=0; i<n; i++){
    printf("%s\n", str[i]);
}
```

Spiral

Date .....

Output:-

Enter number of names:
5

Enter names in any order:
Pinky
Lucky
Ram
Ankit
chaman

The sorted order of names is:
Ankit
chaman
Lucky
Pinky
Ram

Date 13-09-22

5. WAP to implement insertion operation at specified location in an Array.

=> /* insertion operation at specified location in an Array */

```
#include<stdio.h>
int main()
{
    int arr[100] = {0};
    int i, x, pos, n=10;
    For(i=0; i<10; i++)
        arr[i] = i+1;
    For (i=0; i<n; i++)
        printf("%d", arr[i]);
    printf("\n");
    X = 50;
    pos = 5;
    n++;
    For(i=n-1; i>=pos; i--)
        arr[i] = arr[i-1];
    arr[Pos-1] = x;
    For (i=0; i<n; i++)
        printf("%d", arr[i]);
    printf("\n");
    return 0;
}
```

Spiral

Date .....

Output :-

1 2 3 4 5 6 7 8 9 10

1 2 3 4 50 5 6 7 8 9 10

Date 16-09-22

6. WAP to implement deletion operation at specified Location in an Array.

⇒ /* Remove element From array */

```
#include <stdio.h>
int main()
{
  int array[100], position, c, n;
  printf("Enter number of elements in Array\n");
  Scanf("%d", &n);
  printf("Enter %d elements\n", n);
  For (c=0; c<n; c++)
   scanf("%d", &array[c]);
  Printf("Enter the location where you wish to
          delete element\n");
  Scanf("%d", &position );
  if( position >= n+1)
      printf("Deletion is not possible\n");
  else
  {
    for (c = position-1; c<n-1; c++)
       array[c] = array[c+1];
    printf("Resultant Array:\n");
    For (c=0; c<n-1; c++)
       printf("%d\n", array[c]);
  }
  return 0;
}
```

Date .....

Output:

Enter No. Elements in array

5

Enter 5 Elements

4
6
8
10
7

Enter the location where you want to delete element

2

Resultant Array is

4
8
10
7

Date 19-09-22

7. WAP to implement creation of sparse Matrix.

⇒ /* creation of sparse matrix */

```
#include <stdio.h>
#include <stdlib.h>
int main()
{
    int row, col, i, j, a[10][10], count = 0;
    printf("Enter Row\n");
    scanf("%d", &row );
    printf("Enter column\n");
    scanf("%d", &col );
    printf("enter element of Matrix 1\n" );
    For (i=0; i< row; i++){
        For (j=0; j< col; j++){
            scanf("%d", &a[i][j]);
        }
    }
    printf("Elements are: \n" );
    For(i=0; i<row ; i++){
        For (j=0; j< col; j++) {
            printf("%d\t", a[i][j]);
    }
    Printf("\n");
}
/* checking sparse of Matrix */
For (i=0; i< row; i++){
    For(j=0; j< col; j++){
```

Spiral

```
if (a[i][j] ==0)
    count++;
  }
}
  if (count >((row * col)/2))
    printf("Matrix is a sparse Matrix \n");
  else
    printf("Matrix is not sparse Matrix \n");
}
```

Output :-

Run 1:

```
Enter Row : 3
Enter Column : 2
Enter element of Matrix 1
1 0 2 0 2 0
Elements are:
1   0
2   0
2   0
Matrix is not Sparse Matrix
```

Run 2:

```
Enter Row: 3          Enter Column : 2
Enter element of Matrix 1
1 0 0 0 0 0

Matrix is sparse Matrix
```

Date 20-09-22

8. WAP to implement addition of two 2D Arrays.

⇒
```
#include<stdio.h>
int main()
{
int r, c, a[100][100], b[100][100], sum[100][100], i, j;
printf("Enter the number of rows: ");
scanf("%d", &r);
printf("Enter the number of columns");
scanf("%d", &c);
printf("\n Enter elements of 1st matrix:\n");
For (i=0; i<r; ++i)
  For (j=0; j<c; ++j){
    printf("Enter element a%d %d:", i+1, j+1);
    Scanf("%d", &a[i][j]);
  }
printf("Enter elements of 2nd Matrix: \n");
For (i=0; i<r; ++i)
  For (j=0; j<c; ++j){
    printf("Enter Element b%d%d:", i+1, j+1);
    scanf("%d", &b[i][j]);
  }
//adding two matrices
For (i=0; i<r; ++i)
For (j=0; j<c; ++j){
  sum[i][j] = a[i][j] + b[i][j];
}
//printing the result
printf("\n Sum of two Matrices: \n");
For (i=0; i<r; ++i)
  For (j=0; j<c; ++j){
```

Spiral

Date .....

```
        printf("%d", sum[i][j]);
            if(j==c-1){
            printf("\n\n");
        }
    }
    return 0;
}
```

Output :-

```
Enter the number of rows: 2
Enter the number of Columns: 3
Enter the elements of 1st matrix:
Enter element a11: 2
Enter element a12: 3
Enter element a13: 4
Enter element a21: 5
Enter element a22: 2
Enter element a23: 3
Enter the elements of 2nd matrix:
Enter element b11: -4
Enter element b12: 5
Enter element b13: 3
Enter element b21: 5
Enter element b22: 6
Enter element b23: 3
Enter element
Sum of two matrices:
-2      8      7
10      8      6
```

Date 20-09-22

9. WAP to implement multiplication of two 2D Arrays.

```
=> #include<stdio.h>
int main(){
    int a[10][10], b[10][10], c[10][10], n, i, j, k;
    printf("enter the value of N:");
    scanf("%d", &n);
    printf("enter the Elements of matrix-A:\n");
    For (i=0; i<n; i++){
        For (j=0; j<n; j++){
            Scanf("%d", &a[i][j]);
        }
    }
    printf("enter the elements of matrix-B:\n");
    For (i=0; i<n; i++){
        For (j=0; j<n; j++){
            scanf("%d", &b[i][j]);
        }
    }
    For( i=0; i<n; i++){
        For(j=0; j<n; j++){
            c[i][j]=0;
            For(k=0; k<n; k++){
                c[i][j]+=a[i][k]* b[k][j];
            }
        }
    }
    Printf("The product of the two matrices is
            :\n");
```

Date .....

```
For(i=0; i<n; i++){
    For(j=0; j<n; j++){
        printf("%d\t", C[i][j]);
    }
    printf("\n");
}
return 0;
}
```

Output:

```
Enter the value of N : 2
Enter the elements of Matrix - A :
2    2
2    2

Enter the Elements of Matrix - B :
2    2
2    2

Product Of the two Matrices is :
8    8
8    8
```

Date 28-09-22

10. WAP to transpose a 2D Array.

⇒ /* C program to transpose the 2D Array */

```
#include<stdio.h>
void main()
{
    int a[10][10], b[10][10], m,n, i,j;
    printf("\n Enter number of rows & Columns of
            array: ");
    Scanf ("%d %d", &m, &n);
    printf ("\n Enter Elements of 2-D Array:\n");
    For (i=0; i<m; i++)
    {
        For (j=0; j<n; j++)
        {
            Scanf ("%d", & a[i][j]);
        }
    }
    printf("\n\n 2-D Array before transporing:\n\n");
    For(i=0; i<m; i++)
    {
        For (j=0; j<n; j++)
        {
            printf("\t %d", a[i][j]);
        }
        printf ("\n\n");
    }

/* transporting Array */
```

Date .....

```
For(i=0; i<m; i++)
{
    For (j=0; j<n; j++)
    {
        b[j][i] = a[i][j];
    }
}
printf (" \n\n 2-D Array after transposing: \n\n
        ");
For (i=0; i<n; i++)
{
    For (j=0; j<m; j++)
    {
        printf ("\t %d", b[i][j]);
    }
    printf ("\n\n");
}
getch();
}
```

Output:

Enter number of rows & columns of array: 2x2

Enter elements of 2-D Array

12 25 4 62

2-D Array before transposing		2-D After before transposing	
12	25	12	4
4	62	25	62

Date 29-09-22.

11. WAP to implement stack operation using Array

```
#include <stdio.h>
int stack[100], choice, n, top, x, i;
void push (void);
void pop (void);
void display (void);
int main ()
{
    ~~top=1~~;
    top = -1;
    printf ("\n Enter the size of STACK[MAX=
                100]:");
    scanf ("%d", &n );
    printf ("\n\t STACK OPERATIONS USING
                ARRAY");
    printf ("\n\t - - -
```

Date .....

```
    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");
    }
  }
}
  while(choice!=4);
  return 0;
}
Void PUSH()
{
  if (top>=n-1)
  {
    printf("\n\t STACK is over flow");
  }
else
```

Date 06-10-22

```
{
  printf("\n\t Enter a value to be pushed:");
  Scanf("%d", &x);
  top++;
  stack[top]=x;
}
}
Void pop()
{
  if (top<=-1)
  {
    printf("\n\t stack is underflow");
  }
  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
  {
```

Date .....

```
    printf("\n the stack is Empty");
    }
}
```

Output:

```
Enter the size of STACK[MAX=100] : 10
    STACK OPEARATIONS USING ARRAY
    - - - - - - - - - - - - - - - - - -
    1. PUSH
    2. POP
    3. DISPLAY
    4. EXIT
Enter the choice : 1
Enter the value to be pushed : 12
Enter the choice : 1
Enter the value to be pushed : 24
Enter the choice : 1
Enter the value to be pushed : 98
Enter the choice : 3
The elements in STACK
  98    24    12
Press Next choice
Enter the choice : 2
    The popped element is 98
Enter the choice 3
  The elements in STACK
  24    12
Press Next choice
Enter the choice : 4
```

EXIT PATH

Date 06-10-22

12. WAP to implement reverse a string Using stack.

⇒ /* c program to reverse string Using STACK */

```
#include<stdio.h>
#include<string.h>
# define MAX 100
  int top=-1;
  int item;
  char stack_string [MAX];
  void pushchar (char item);
  char popchar(void);
  int isEmpty(void);
  int isFull (void);
  int main()
  {
    char str[MAX];
    int i;
    printf ("Input a string");
    Scanf ("%[^\n]s", str);
    For(i=0; i<strlen (str); i++)
        pushchar(str[i]);
    For(i=0 ; i<strlen (str); i++)
      str[i] = popchar();
    printf(" Reversed string is: %s \n", str);
    return 0;
  }
Void pushchar (char item)
{
  if (isFull())
```

Date .....

```
{
  printf("\n stack is Full!!!\n");
  return;
}
  top = top+1;
  stack_string[top] = item;
}
char popchar()
{
  if(isEmpty())
  {
    printf("\n stack is empty!!! \n");
    return 0;
  }
  item = stack_string[top];
  top = top-1;
  return item;
}
int isEmpty()
{
    if(top==-1)
      return 1;
    else
        return 0;
}
int isfull()
{
    if(top=MAX-1)
      return 1;
    else
  return 0;
```

Date .....

}

Output:

Input a string: Hello world!

Reversed string is : !dlrow olleH

Seen
Kaiw
21/11/22

Date .....

Que 13. Implementing stack using Arrays :-

```
// c program for array implementation of stack

#include <conio.h>
#include <stdio.h>
#include <stdlib.h>
// A structure to represent a stack
 struct stack {
   int top;
   Unsigned capacity;
   int* Array;
   };
   Struct stack* CreateStack (Unsigned capacity)
   {
     Struct stack* stack = (struct stack*) malloc (
          size of (struct stack));
     Stack -> capacity = capacity;
     Stack -> top = -1;
     Stack -> Array = (int *) malloc(stack-> capacity *
          sizeof (int));
     return stack;
   }
   int isfull (struct stack* stack)
   { return stack -> top == stack -> capacity - 1;
   } int isEmpty (struct stack* stack) {
     return stack->top == -1;
   } void push (struct stack* stack, int item)
   { if (isFull (stack))
        return;
        stack -> Array[++stack -> top] = item;
```

Date .....

```
printf("%d pushed to stack\n", item);
}
 int pop (struct stack* stack)
 { if (isEmpty (stack))
      return INT_MIN;
   return stack -> Array[stack -> top--];
 int peek (struct stack* stack)
 { if (isEmpty (stack))
      return INT_MIN;
      return stack -> array[stack -> top];
 } int main()
 { struct stack* stack = Create Stack(100);
   push (stack, 10);
   push (stack, 20);
   push (stack, 30);
   printf("%d popped from stack\n", pop (stack));
   return 0;
 }
```

Output :-

```
10 pushed into stack
20 pushed into stack
30 pushed into stack
30 popped into stack
Top element is: 20
Elements present in stack: 20 10
```

Date .....

Que 14. Implementing stack using Linked list:

/c program for linked list implementation of stack

```
#include <stdio.h>
#include <stdlib.h>
Struct StackNode {
 int data;
 Struct StackNode* next;
};
Struct StackNode* newNode (int data)
{ Struct StackNode* stackNode = (struct StackNode*)
   Malloc (sizeof (Struct stackNode));
   StackNode -> data = data;
   StackNode -> Next = NULL;
   return StackNode;
} int isEmpty (struct StackNode* root)
 { return! root;
 } void push (Struct StackNode** root, int data)
 { Struct StackNode* stackNode = newNode (data);
   StackNode -> next = *root;
   *root = stackNode;
   printf ("%d pushed to stack \n", data);
 } int pop (Struct StackNode** root)
 { if (isEmpty(*root))
       return INT_MIN;
   Struct StackNode* temp = *root;
   *root = (*root) -> next;
   int popped = temp -> data;
   free (temp);
   return popped;
```

Date .....

```
} int peek (struct StackNode* root)
{ if (isEmpty(root))
      return INT_MIN;
  return root -> data;
} int main()
{ struct StackNode* root = NULL;
  push(&root, 10);
  push(&root, 20);
  push(&root, 30);
  printf("%d popped from stack\n", pop(&root));
  printf("Top element is %d\n", peek(root));
  return 0;
}
```

Output:-

```
10 pushed to stack
20 pushed to stack
30 pushed to stack
30 popped from stack
Top element is 20
Elements present in stack : 20  10
```

Date .....

(15.) Singly linked list :-

```
// Function to create list with n nodes initially

#include<stdio.h>
#include<stdlib.h>
//Linked list node
 Struct node {
   int info;
   Struct node* link;
};  Struct node* Start = NULL;
 Void CreateList()
 { if (start == NULL) {
     int n;
     printf("\nEnter the number of nodes:");
     Scanf("%d", &n);
     if (n != 0) {
     int data;
     Struct node* newnode;
     Struct node* temp;
    newnode = Malloc (sizeof (Struct node));
     Start = newnode;
     temp = Start;
     printf("\nEnter number to" "be inserted:");
     Scanf("%d", &data);
     Start -> info = data;
     For (int i=2; i<=n; i++)
      newnode = malloc (sizeof (Struct node)
       temp -> link = newnode;
     printf("\n Enter number to" "be inserted:");
      Scanf("%d", &data);
    newnode -> info = data;
```

Date .....

```
      temp = temp -> link;
   }} printf("\n the list is created \n");
   } else
      printf("\n The list is already created \n");
   }
//Function to traverse the linked list
 Void traverse()
 { Struct node* temp;
   // List is empty
   if (start == NULL)
      printf("\n List is empty \n");
   //Else print the LL
   Else {
      temp = start;
      while (temp != NULL) {
      printf("Data = %d \n", temp -> info);
      temp = temp -> link;
    }}}
//Driver code.
 int main()
 { int choice;
   while (1) {
      printf("\n\t 1 To see list \n");
      printf("\n\t 2 For insertion at Starting \n");
      Printf("\t3 For insertion at End \n");
      Printf("\t4 for insertion at any position \n");
      printf("\t5 For deletion of first element \n");
      printf("\t6 for deletion of last element \n");
      printf("\t7 for deletion at any position \n");
      printf("\t8 To find Maximum among the elements \n"
      printf("\t9 to find Mean of the elements \n");
```

Date .....

```
printf("\t10 To sort element\n");
printf("\t11 To reverse the linked list\n");
printf("\t12 To exit\n");
printf("\n Enter Choice:\n");
Scanf("%d", &choice);
Switch(choice){
Case 1:
    traverse();
    break;
Case 2:
    insert-At-End();
    break;
Case 3:
    insert Affront();
    break;
Case 4:
    insertAtPosition();
    break;
Case 5:
    deleteFirst();
    break;
Case 6:
    deleteEnd();
    break;
Case 7:
    deletePosition();
    break;
Case 8:
    Maximum();
    break;
Case 9:
```

Date .....

```
mean();
break();
Case 10:
    sort();
    break;
Case 11:
    reverseLL();
    break;
    ~~Case 12:~~
Case 12:
    exit(1);
    break;
default:
    printf("Incorrect choice\n");
}
}
return 0;
}
```

Output:- 1. To see list
2. For insertion at starting
3. For insertion at end
4. For insertion at any position
5. For deletion of first element
6. For deletion of last element/at any position
7. to find maximum among the elements
8. To find mean of the elements
9. To sort element
10. To reverse the linked list
11. To exit

Enter choice:

Date .....

(16.) Function to insert / delete at the front of the linked list

(insertion)

```
Void insertAtFront()
{
  Int data;
  Struct node * temp;
  temp = malloc (sizeof (struct node));
  printf ("\n Enter number to be inserted:");
  scanf ("%d", &data);
  temp -> info = data;

  // Pointer of temp will be assigned to start
  temp -> link = start;
  start = temp;
}
```

(Deletion)

```
void deletefirst()
{
  struct node * temp;
  if (start = = NULL)
    printf ("\n List is empty\n");
  else {
    temp = start;
    Start = start -> link;
    free (temp);
  }
}
```

Date .....

(17.) Insertion / deletion at the End :-

(Insertion)

```
Void insertAtEnd()
{
  int data;
  Struct node *temp, *head;
  temp= malloc (size of (struct node));
  // Enter the number
  printf ("\n Enter number to be inserted : ");
  scanf ("%d", &data);
  //changes links
  temp-> link = 0;
  temp-> info = data;
  Head = start ;
  while (head -> link != NULL){
      head = head -> link ;
  }
   head -> link = temp;
}
```

(Deletion)

```
Void deleteEnd()
{
  Struct node *temp; *prevnode;
  if(start ==NULL)
     printf("\n list is Empty\n");
    else {
     temp = Start ;
    while (temp -> link != 0){
     Prevnode = temp;
```

Date .....

```
    temp = temp->link;
  }
    free(temp);
    prevnode->link = 0;
  }
}
```

Date .....

(18.) Insertion/Deletion From any specified position

(Insertion)

```
Void insertAtPosition()
{
  struct node *temp, *newnode;
  int Pos, data, i=1;
  newnode = malloc (sizeof (struct node));

  //Enter the position and Data
  Printf("\n Enter Position and Data:");
  Scanf("%d %d", &Pos, &Data);
  //change links
  temp = start;
  newnode -> info = data;
  newnode -> link = 0;
  while (i< pos-1) {
    temp = temp-> link;
    i++;
  }
  newnode-> link = temp -> link;
  temp -> link = newnode;
}
```

(Deletion)

```
Void delete position()
{
  struct node * temp, * Position;
  int i=1, Pos;
```

Date .....

```
//if LL is empty
if (Start == NULL)
  printf("\n List of empty \n");

//otherwise
else {
    printf("\n Enter Index");

//Position to be deleted
Scanf("%d", &pos);
position = malloc(sizeof(struct node));
temp = start;

//Traverse till position
while (i< pos-1) {
  temp = temp-> link;
  i++;
}

//change Links
position = temp -> link;
temp -> link = position -> link;

//Free memory
Free (position);
}
}
```

Date .....

(20.) Function to find the maximum element in the linked list

```
Void maximum()
{
    int a[10];
    int i;
    struct node* temp;

// If LL is empty
    if (start == NULL)
        printf("\n List is empty\n");

// otherwise
    Else {
        temp = start;
        int max = temp -> info;
// Traverse LL and update the maximum element

        while (temp != NULL) {

// update the maximum element

        if (max < temp -> info)
            (max = temp -> info);
        temp = temp -> link;
    }

    Printf("\n Maximum number is: %d", max);
  }
}
```

Date .....

(21.) Function to sort the linked list in ascending order.

```
Void sort()
{
  Struct node* current = start;
  Struct node* index = NULL;
  int temp;

  // If LL is empty
    if (start == NULL){
      return;
    }

  // Else
    else {
      // Traverse the LL
      while (current != NULL) {
          index = current->link;

  // Traverse the LL nestedly and find the
     minimum element
  while (index != NULL){

  // swap with it the value at current

  if (current->info > index->info){
      temp = current->info;
      current->info = index->info;
      index->info = temp;
  }
  index = index->link;
```

Date .....

```
    }

    // update the current
    current = current -> link;
    }
  }
}
```

Date .....

(22.) Reverse the linked list :-

```
Void reverseLL()
{
  struct node *t1, *t2, *temp;
   t1 = t2 = NULL;

  // If LL is empty
  if (start == NULL)
     printf ("List is empty \n");

// else
  else{

    // Traverse the LL
      while (start != NULL){

        //reversing of pointe
        t2 = start -> link;
         start -> link = t1;
         t1 = start;
         Start = t2;
      }
       Start = t1;
      // new head note
        temp = start;

      printf ("Reversed linked list is ");

      //print the LL
```

Date .....

```
while (temp != NULL) {
        printf ("%d", temp→info);
        temp = temp→ link;
    }
  }
}
```

Date .....

Que 23. Insert and delete a node in DOBLY Linked list.

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

//node creation
 Struct node {
  int data;
   Struct ~~dat~~ Node* next;
   Struct Node* prev;
  };

//insert node at the front

Void insertFront (Struct Node ** head, int data) {

// Allocate memory for new node

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

//assign data to newNode
newNode→ data = data;

//make newNode as a head
newNode→ next = (*head);

//assign null to prev
newNode → prev = NULL;
```

Date .....

```
//previous of head (now head is the second
node) is newNode

if((*head)!=NULL)
(*head)→Prev = newNode;

//head points to newNode
(*head)=newNode;
}

//insert a node after a specific node
void insertAfter(struct Node* prev_node, int data){

//check if previous node is ≠ null

if(prev_node==NULL){
printf("previous node cannot be null");
return;
}
//allocate memory for newNode

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

// assign ~~to~~ data to newNode
~~Struct Node* newNode = (struct Node*) malloc~~
newNode→data = data;

//set next of newNode to next of prev node
newNode→next = prev_node→next;
```

Spiral

Date .....

```
//set next of prev node to newNode
prev_node→next = newNode;

//set prev of newNode to the previous node.
newNode→prev = prev_node;

//set prev of newNode's next to newNode
if(newNode→next!=NULL)
newNode→next→prev = newNode;
}

//insert a newNode at the end of the list void
insertEnd(struct Node** head, int data){

//allocate memory for node.
struct Node* newNode = (struct Node*) malloc(sizeof
(struct Node));

// assign data to newNode
newNode→data = data;

//assign null to next of newNode
newNode→next = NULL;

//store the head node temporarily (for later use)
struct Node* temp = *head;

//if the linked list is empty, make the newNode as
a head node.
if(*head==NULL){
newNode→Prev = NULL;
```

Spiral

Date .....

```
*head = newNode;
  return;
}

// if the linked list is not Empty, traverse to the
End of the linked list
while (temp->next != NULL)
  temp = temp->next;

// now, the last node of the linked list is temp

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

Void deleteNode (struct Node** head, struct Node* del_
node){
  if (*head == NULL || del_node == NULL)
  return;

if (*head == del_node)
  *head = del_node->next;

del_node
  if (del_node->next != NULL)
  del_node->next->prev = del_node->prev;

del_node
  if (del_node->prev != NULL)
    del_node->prev->next = del_node->next;
```

Date .....

```
free(del_node);
}
Void displayList (struct Node* node){
  Struct Node* last;

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

int main (){
  Struct Node* head = NULL;

  insertEnd(&head, 5);
  insertFront (&head, 7);
  insertFront (&head, 6);
  insertEnd (&head, 9);

  insertAfter (head, 11);
  insertAfter (head->next, 15);
  displayList (head);
  deleteNode (&head, head->next->next->next);

  displayList (head);
}
```

Date .....

Que 24. C Program For implementation of selection sort

```
#include <stdio.h>

Void swap (int *xp, int *yp)
{
  int temp = *xp;
  *xp = *yp;
  *yp = temp;
}
Void selectionSort (int arr[], int n)
{
  int i, j, min_idx;

  //one by one move boundary of unsorted subarray
  For(i=0; i<n-1; i++)
  {

  //Find the minimum Element in unsorted array
    min_idx = i;
    For (j=i+1; j<n; j++)
    if (arr[j]< arr[min_idx])
      min_idx = j;
    Swap (&arr[min_idx], &arr[i]);
  }
}

Void printArray(int arr[], int size)
{
  int i;
  For(i=0; i<size; i++)
```

Spiral

Date .....

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

//Driver program to test above functions

int main()
{
    int arr[] = {64, 25, 12, 22, 11};
    int n = sizeof(arr)/sizeof(arr[0]);
    selection sort (arr, n);
    printf("Sorted Array:\n");
    printArray(arr, n);
    return 0;
}
```

Date .....

Que 25. C program for insertion sort

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

Void insertion sort (int arr[], int n)
{
 int i, key, j;
 For (i=1; i<n; i++)
 {
  key= arr[i];
  j= i-1;
  while (j>=0 && arr[j]>key)
  {
   arr[j+1]=arr[j];
   j=j-1;
  }
  arr[j+1]=key;
 }
}

Void printArray (int arr[], int n)
{
 int i;
 for (i=0; i<n; i++)
    printf ("%d", arr[i]);
 printf("\n");
}

int main()
```

Date .....

```
{
 int arr[]={12,11,13,5,6};
 int n = sizeof(arr)/sizeof(arr[0]);
 insertion sort (arr,n);
 printArray(arr,n);
 return 0;
}
```

Spiral

Spiral

Date .....

Que 26. C program for implementation of Bubble Sort

```
#include<stdio.h>
void swap (int *xp, int *yp)
{
int temp = *xp;
*xp = *yp;
*yp = temp;

void bubblesort (int arr[], int n)
{
int i,j;
For (i=0;i<n-1;i++)

For (j=0; j<n-i-1;j++)
if (arr[j] > arr[j+1])
Swap (&arr[j], &arr[j+1]);
}

void printArray (int arr[], int size)
{
Int i;
For (i=0; i<size; i++)
printf ("%d ", arr[i]);
printf("\n");

int main()
{

int arr[] = {64, 34, 25, 12, 22, 11, 90};
```

Date .....

```
int n = sizeof(arr)/sizeof(arr[0]);
bubblesort (arr, n);
printf("Sorted Array:\n");
PrintArray (arr, n);
return 0;
```

Date .....

Ques: C program for merge sort.

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

void merge (int arr[], int l, int m, int r)
{
    int i,j,k;
    int n1 = m-l+1;
    int n2 = r-m;
    int L[n1], R[n2];
    for (i=0; i<n1; i++)
    L[i] = arr[l+i];
    For (j=0; j<n2; j++)
    R[j] = arr[m+1+j];
    i = 0;
    j = 0;
    k = l;
    {
    while (i<n1 && j<n2)
    {
     if (L[i]<=R[j])
     {
      arr[k] = L[i];
      i++;
     }
     else
     {
      arr[k] = R[j];
      j++;
     }
```

Date .....

```
k++;
}
while (i<n1){
    arr[k] = L[i];
    i++;
    k++;
}
while (j<n2)
{
    arr[k] = R[j];
    j++;
    k++;
}
}
void mergeSort (int arr[], int l, int r)
{
    if (l<r)
    {
      int m = l+(r-l)/2;
      mergeSort (arr, l, m)
      mergeSort (arr, m+1, r);
      merge (arr, l, m, r);
    }
}

//Utility Functions Functions to print an array

void printArray(int A[], int size)
{
    int i;
```

Spiral

Date .....

```
For (i=0; i< size; i++)
    printf ("%d", A[i]);
    printf ("\n");
}
int main()
{
    int arr[]= {12,11,13,5,6,7};
    int arr_size = size of (arr)/ sizeof (arr[0]);
    printf ("Given array is \n");
    printArray(Arr, arr_size);

    mergeSort(arr,0, arr_size-1);
    printf ("\n SortedArray is \n");
    printArray(arr, arr_size);
    return 0;
}
```