VIT®
BHOPAL
www.vitbhopal.ac.in

Programme: B. Tech in CSE (Specialization)

Course Code and Title: CSE2002 - Data Structures and Algorithms

Assignment – 1

Submitted by: Harsh Raj Verma
Registration No: 21BCE11028

Submitted to: Dr. Paras Jain
Senior Assistant Professor SCSE

INDEX

S No.	Name of Experiment	Date of Experiment	Date of Submission	Page No.	Remarks
Module 1					
1	FactorialRec.c	02/2/22	27/3/22		
2	FactorialNonRec.c	02/2/22	27/3/22		
3.	TowerofHanoi.c	02/2/22	27/3/22		
4.	BinNonRec.c	04/2/22	27/3/22		
5.	BinRec.c	04/2/22	27/3/22		
6.	RecMaxElement.c	07/2/22	27/3/22		
7.	MaxElement.c	04/2/22	27/3/22		
8.	UniqueElements.c	07/2/22	27/3/22		
9.	MatrixMul.c	04/2/22	27/3/22		
Module 2					
10.	SequentialSearch.c	28/01/22	27/3/22		
11.	SelectionSort.c	14/2/22	27/3/22		
12.	BubbleSort.c	14/2/22	27/3/22		
13.	Insertionsort.c	16/2/22	27/3/22		
14.	MergeSort.c	21/2/22	27/3/22		
15.	QuickSort.c	21/2/22	27/3/22		
16.	RecBinSearch.c	23/2/22	27/3/22		
17.	BinSearch.c	23/2/22	27/3/22		
Module 3					
18.	LinkedList.c	21/3/22	27/3/22		
19.	SingleLinkedlist.c	02/3/22	27/3/22		
20.	DoubleLinkedList.c	04/3/22	27/3/22		
21.	CircularSingleLL.c	14/3/22	27/3/22		
22.	CircularDoubleLL.c	16/3/22	27/3/22		
23.	PolyAddition.c	23/3/22	27/3/22		
24.	PolyLinkedList.c	23/3/22	27/3/22		

MODULE - 1

PROGRAM - 1 Sequential Search :-

```
#include <stdio.h>
int sequential srch (int [], int, int);
main ()
{
    Printf (" Program for sequential Search using
            Brute force Approach \n");
    Print f (" Author : Harsh Raj \n");
    Print f (" Experiment Date : 31-01-2022 \n");
    print f (" submission date : 27-03-2022 \n");
```

```
int A[50], key, n, i, pos;
Printf(" Enter the array size : ");
Scanf("%d", &n);
Printf(" Enter the array elements : \n");
for (i = 0; i <= n-1; i++)
{
    Printf(" Enter the %d element:
    Scanf(" %d ", &A[i]);
}
Printf(" Enter the key to be searched : ");
scanf("%d", &key);
Pos sequential Srch (A, key, n);
if [pos >= 0 && pos <= n-1)
{
    Printf(" The key is found in array at location
                %d ", pos+1);
}
else
{
    Printf("The key is not found in array");
}

int sequential srch (int A[], int key, int n)
{
    int i = 0
    while (i < n && A[i] != key)
    {
        i = i+1;
    }
    if (i < n)
    {
```

PROGRAM - 2 Factorial Rec.c

```
#include <stdio.h>
int factorial (int);
main()
{
    Printf("program for factorial of a non-negative
            integer with Recursive function \n");
    printf(" Sumitted : Harsh Raj \n");
    printf(" Experiment date : 02-02-2022 ");
    printf(" submission date : 27-03-2022 \n");

    int num, fac;
    printf ("Enter the number ");
    scanf("%d ", &num);
    fac = factorial (num);
    printf ("\n the factorial of %d is %d ,num
}

int factorial (int n) {
  if /*n*/ !=0
  if (n==0
  return 1;
  else
  return n* factoria (n-1);
```

PROGRAM-3 Factorial Non Rec.c

```
# include <Stdio.h>
int Non Rec factorial (int);

main()
{
  Printf (" program for factorial of a non-negative
          integer with Non Recursive function \n");
  Printf (" submitted by: Harsh Raj \n");
  printf (" Experiment date : 02-02-2022 \n");
  Printf (" Submission date : 27-03-2022 \n \n");

  int num, fac;
  printf ("Enter the number : ");
  Scanf ("%d", &num);
  fac = Non Rec factorial (num);
  Printf ("\n The factorial of %d is %d \n", num
                                        , fac);
}
int Non Rec factorial (int n)
{
  int i, fact = 1;
  for (i=1; i<=n; i++)
  {
    fact = fact * i;
  }
  return fact;
}
```

PROGRAM - 4 Tower of Hanoi - C

```
# include <Stdio.h>
void Tower-of-Hanoi (int, char, char, char);
int pow (int, int);

main ()
{
  int n;
  int p;
  Printf ("program to find moves for ToH puzzle\n");
  printf ("submitted by: Harsh Raj \n");
  Printf ("Experiment date : -02-2022\n");
  Printf ("Enter the number of discs :");
  Scanf ("%d", &n); //input for no discs
  Tower-of-Hanoi (n, 'S', 'A', 'D');
  Printf ("Total no. of disks moved := %d", pow(2,n)-1
}

// This function prints the required moves of disks
void Tower-of-Hanoi (int n, char s, char A, char D)
{
  if (n==1)
  {
    Printf ("Move disk from %c to %c \n", s, D);
    return 0;
  }
  else
  {
    Tower-of-Hanoi (n-1, S, D, A);
    Print f("Move disk from %c to %c \n", S, D);
  }
}
```

```
int pow(int x, int y)
{
    int res = 1;
    for (int i = 1; i <= y; i++)
    {
        res = res * x;
    }
    return res;
}
```

5. BinNonRec.c

```
#include <stdio.h>

int NonBinRec(int);

main()
{
    Printf("Program to determine no. of bits to
            represent a given decimal no. in
            binary from using NonBinRec\n");

    ~~Printf("Author : Dr/Prof Harsh/Raj/\n");~~
    Printf("Author : Dr paras Jain \n");
    Printf("Experiment date : 04-02-2022 \n");
    Printf("submission date : 27-02-2022 \n");

    int n, bits;
    printf("Enter the decimal number : ");
    Scanf("%d", &n);
```

```
bits = NonBinRec(n);
Printf("\n %d bits are required to represent
        the decimal number %d in binary form",
        bits, n);

int NonBinRec(int n)

{
int count = 1;
While (n>1)
  {
    count = count + 1;
      n = n/2
  }
return count;
```

6. PROGRAM-6 BinRec.c

```
#include <Stdio.h>
int BinRec(int);
main():
  {
  Printf(" program to determine the number of bits
          to request a given decimal in binary form");
  Printf("Submitted by: Harsh Raj");
  Printf("Experiment date: 04-02-2022");
  Printf("Submission date: 17-03-2022");
  int n, bits;
  Printf("Enter the ~~deter~~ decimal Number");
  Scanf("%d", &n)
  bits = BinRec(n);
```

```
    Printf ("\n%d bits required to represent decimal
            no. %d in binary form", bits, n);
}
int BinRec (int n)
{
if (n == 1 || n == 0
{
}
return 1;
}
```

7. PROGRAM :- Rec Max Element - C

```
#include <stdio.h>
int Rec Max Element (int [], int);
int Max (int, int);

main()
{
    Printf ("program to find largest Element \n");
    Printf ("Submitted by: Harsh Raj \n");
    Printf ("Experiment date : 04-02-2022 \n");
    Printf ("Submission date : 27-03-2022. \n");

    int A[50], n, i, marval;
    Printf ["Enter the Array Size : ");
    scanf ("%d", &n)
    Printf ("Enter the Array elements : ");
    for (i = 0; i <= n-1; i++)
    {
        Printf ("Enter the element number %d :", i+1);
```

```
    Scanf ("%d", &A[i]);
}
Printf("\n The elements entered for Array:/n");
Printf ("[");
for (i=0; i<=n-1; i++)
{
Printf ("%d", A[i]);
}
Printf ("]")
Maxval = Rec Max Element (A,n);
Printf("\n\n The largest element in Array is %d"
                                    maxval);
}
```

The function Rec Max Element finds largest element in a given input Array of n size by Recursion

```
int Rec Max Element (int A[ ], int n)
{
  If (n==1)
  {
    return A[0];
  }
  else
  {
    return max (Rec Max Element (A, n-1), A[n-1]);
  }
}
```

8. PROGRAM – 8 Max Element.C

```
# include <stdio.h>
int Max Element (int[ ], int);
main()
{
    Print f(" program to find largest element in array \n");
    print f (" project by : Harsh Raj \n");
    Print f (" Experiment date : 04-02-2022 );
    Print f (" Submission date : 27-03-2022 ");
    int A[50], n, I , maxval;

    Print f (" Enter the Array size ");
    Scan f (" % d", & n);
    Print f (" Enter Array Elements : \n");
    for (i = 0; i <= n-1; i++ )
    {
    Print f (" Enter the element number % d : ", i+1);
    }
    Print f (" \n The elements entered for the array
                are \n ");
    Print f (" [ ");
    for (i = 0, I <= n-1; i++);
    {
    Print f (" % d ", A[i ]);
    }
    Print f ( " ] ");
    Max val = Max Element (A, n);
    Print f (" \n \n The largest element in Array % d", max
                val);
```

```
int Max Element (int A[], int n)
{
int Maxval, i;
maxval = A[0];
for (i = 1; i <= n-1; i++)
{

if [A[i] >= maxval
{
maxval = A[i];
}
}
return maxval;
}
```

9. PROGRAM-9 Unique Element.c

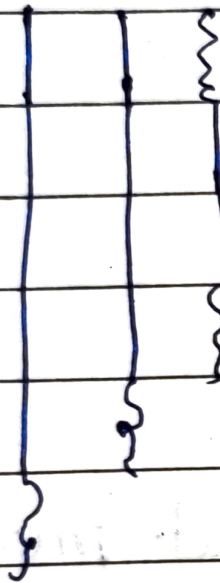
```
#include <stdio.h>
int Unique Elements (int [], int);

main()
{
Print f(" Program to find if elements are unique \n");
Print f(" submitted by: Harsh Raj "\n);
Print f("Experiment date: 04-02-2022 \n \n");
int A[50], n i, flag;

Print f(" Enter the Array size: ");
Scan f("%d", & n);
Print f(" /n Enter the ~~Array~~ array elements: \n");
```

```
for (i=0; i<=n-1; i++)
{
Printf("Enter the element no. %d: ", i+1);
Scanf("%d", &A[i]);
}
Printf("\n The elements entered in the array are: \n");
Printf("[");
for (i=0; i<=n-1; i++)
{
Printf("%d", A[i]);
}
Printf("]\n")
flag = Unique Elements(A,n);
if (flag==1)
Printf("\n The elements in array are unique");
else
Printf("\n The elements in array aren't unique");
}
int Unique Elements(int A[], int n)
{
int i, j;
for (i=0; i<=n-2; i++)
{
for (j=0; j<=n-1; j++)
{
if (A[i]==A[j])
return 0;
}
}
return 1;
}
```

10. PROGRAM-10 Matrix Mul.c

```
#include <stdio.h>
void MatrixMul(int [][20], int [][20], int);
void printArray(int A[][20], int n);

main()
{
  int A[20][20], B[20][20], n, i, j;
  Printf("program for multiplication of 2 matrices\n");
  Printf("Submitted by: Harsh Raj\n");
  Printf("Experiment date: 04-02-2022");
  Printf("Submission date: 27-03-2022 );
  Printf("_____");
  Printf("\n Enter the Matrix dimension: ");
  Scanf("%d", &n);
  for (i=0; i<=n+; i++)
  {
    for (j=0; j<=n-1; j++)
    {
      Printf("Enter the input for A[%d][%d]: ", i, j);
      Scanf("%d", &A[i][j]);
    }
  }
  Printf("\n");
  Printf("The input matrix A of %d * %d is :\n", n, n);
  Print Array(A, n);
  Printf("\n");

  Printf("\n The input Matrix B of %d * %d is :\n", n, n);
```

```
Print Array (B, n);
Printf ("\n");
Matrix Mul (A, B, n);
}

// This function computes multiplication of input matrics A & B
void Matrix Mul (int A[][20], int B[][20], int n)
{
    int C[20][20], i, j, k;
    for (i=0; i<=n-1; i++)
    {
        for (j=0; j<=n-1; j++)
        {
            C[i][j] = 0;
            for (k=0; k<=n-1; k++)
            {
                C[i][j] = C[i][j] + A[i][j] * B[i][j];
            }
        }
    }
    Printf ("The output C=A* B is as follows :\n")
    print Array (C, n);
}

// This function prints element in Matrix
void print Array (int A[][20], int n)
{
    int i, j;
    for (i=0; i<=n-1; i++)
    {
        for (j=0; j<=n-1; j++)
```



```
Print f ("%d", A[i][j]);
Print f (" ");
```

MODULE - 2

* PROGRAM - 1 Selection.c

```
#include <stdio.h>
void selection_sort(int A[], int n);
void printArray(int A[], int n);
main()
{
  int n, A[20], i;

  Printf("program for sorting an array using iterative form
        based selection sort \n");
  Printf("Submitted by: Harsh Raj \n");
  Printf("Experiment date: 14-02-2022 \n");
  Printf("\n Enter the array size: ");
  Scanf("%d", &n);
  for (i=0; i<=n-1; i++)
  {
    Print f("Enter the element number %d: ", i+1);
    Scan f("%d", &A[i]);
  }

  Print f("\n The array before sorting is :[");
  Print Array (A, n);
  Print f("]\n\n")
```

```
selection_sort (A, n);
Printf ("The array after sorting is : [");
PrintArray (A) n);
Printf ("]\n\n");

// This function prints the elements of an array
void printArray (int A[], int n)
{
    int i;
    for (i = 0; i <= n-1; i++)
    {
        Printf ("%d", A[i])
        Printf ("  ");
    }
```

PROGRAM - 2 Bubble.c

```
# include <stdio.h>
Void Bubble_sort (int A[], int n);
Void Print Array (int A[], int n);

main ()
{
    int A[20], n, i;
    Printf ("program for sorting an array using
            brute force based bubble sort\n");
    Printf ("Author : Harsh Raj \n");
    Printf ("Expriment date : 14-02-2022\n");
    Printf ("Submission date : 27-03-2022 \n\n");
    Printf ("- - - - - - - - - - - - - - - -");
```

```
Printf("Enter Array size : ");
Scanf("%d", &n);
Printf("\n");
for (i=0; i<=n-1; i++)
{
    Printf("Enter the element no.%d :", i+1);
    Scanf("%d", &A[i]);
}
Printf("\n The array before sorting is :[ ");
Print Array (A, n)
Printf("]\n\n");

Bubble_sort (A, n);
Printf("The array after sorting is : [ ");
Print Array (A, n);
Printf(" ]\n\n");
}

// This functions sorts elements in ascending order
Void Bubble_Sort(int A[], int n
{
    int i, j, temp;
    for (i=0; i<=n-2; )
    {
        for (j=0; j<=n-2-i; j++ )
        {
            if (A[j+1] < A[j])
            {
                temp = A[j];
                A[j] = A[j+1];
                A[j+1] = temp;
            }
```

```
    }
}
// This function prints the elements in an array.
void printArray(int A[], int n);
{
    int i;
    for (i=0; i<=n-1; i++)
    {
        Printf("%d", A[i]);
        Printf(" ");
    }
}
```

3. PROGRAM -3 Insertion:

```
#include<stdio.h>
void insertion_sort(int A[], int n)
Void printArray(int A[], int n);
main()
{
    int A[20], n, i
    printf("program for sorting an array using brute
            force based insertion sort \n");
    printf("Author : Harsh Raj \n");
    Printf("Submission date : 27-03-2022 \n\n");
    Printf("_____");
    Printf("\n Enter the array size : ");
    Scanf("%d", &n);
    Printf("\n");
    for (i=0; i<=n-1; i++)
```

```
{
    Printf("Enter the element no.%d: ", i+1);
    Scanf("%d", &A[i]);
}

Printf("The array before sorting is: [");
Print Array(A,n);
Printf("]\n\n");

insertion_sort(int A, n);
Printf("The array after sorting is: [");
Print Array(A,n);
Printf("]\n\n");
```

```
void insertion_sort(int A[], int n)
{
    int key, j, i;
    for(j=0; j<=n-1; j++)
    {
        key = A[j];
        i = j-1

        // Insert A[j] in sorted sequence [A[0]....j-1]
        while(i>=0 && A[i]>key)
        {
            A[i+1] = A[i]
            i = i-1;
        }
    A[i+1] = key;
    }
}
```

* PROGRAM-4 Mergesort.c

```
#include <stdio.h>
void Mergesort (int A[ ], int n);
void Merge (int B[ ], int C[ ], int A[ ], int p, int q);
void printArray (int A[ ], int n);

main()
{
    int A[20], n, i;
    Printf ("program to sort an array using divide and
    Conquer based merge sort \n");
    Printf ("Author : Harsh Raj");
    Printf ("Experiment date : 19-02-2022\n");
    Printf ("Submission date : 27-03-2022 \n");
    Printf ("_ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _");
    Printf ("\n Enter the array size");
    Scanf ("%d", &n);
    Printf ("\n");
    for (i=0; i<=n-1; i++)
    {
        Printf ("Enter the element no. %d: ", i+1);
        Scanf ("%d", &A[i]);
    }

    Printf ("The Array before sorting is : [");
    Print Array (A, n);
    Printf ("]\n\n");

    Merge sort (A, n);
    Printf ("The array after Sorting is: [");
```

```
    Print Array (A, n);
    Printf ("]\n\n")
}
// This function sorts the elements of array in ascend
   order
Void Merge Sort (int A[], int n)
{
    int s, i, j = 0, k, B[20], C[10], p, q;
    if (n>1)
    {
        for (i=0 ; i<= n/2 -1; i++)
        {
            B [i] = A [i];
        }
        P= i;
        for (k= n/2 ; k<=n-1 ; k++)
        {
            C [j] = A [K];
            j++;
        }
        q =j;
        Merge sort (B, P);
        Merge sort (C, q);
        Merge (B, C, A, P, q);
    }
}
// This function merged two sorted arrays
void Merge [int B[], int C[], int A[], int p, int q]
{
[int i= 0, j= 0, k= 0, int ], int 2 ;
```

```
while (i < p && j < q)
{
    if (B[i] <= C[j])
    {
        A[k] = B[i]
        i = i + 1;
    }
    else
    { A[k] = C[j];
      j = j + 1;
    }
    k = k + 1;
}
ind2 = k;
if (i == p)
{
    for (ind1 = j; ind1 < q-1; ind1++)
    {
    A[ind2] = C[ind1];
      ind2 = ind2 + 1;
    }
}
else
{
    for (ind1 = i; ind1 <= p-1; ind1++)
    {
        A[ind2] = B[ind1];
        ind2 = ind2 + 1;
    }
}
}
```

* PROGRAM-5 Quicksort.c

```
#include <stdio.h>
void Quicksort (int A[], int p, int r);
int partition (int A[], int p, int r);
void exchange (int *x, int *y);
void printArray (int A[], int n);

main()
{
  int A[20], n, i;
  Printf("program for sorting an array using div
        and Conquer Quicksort \n");
  Printf("Author : Harsh Raj");
  Printf("Experiment date : 19-02-2022\n");
  Printf("Submission date : 27-03-2022 \n");
  Printf("---------- ---------- ----------");
  Printf("\n Enter the array size : ");
  Scanf("%d", &n);
  Printf("\n");
  for(i = 0; i <= n-1; i++)
  {
    Printf("Enter element no. %d : ", i+1);
    Scanf("%d", &A[i]);
  }
  Printf("The Array before sorting is :[");
  PrintArray(A, n);
  Printf("]\n\n");
}
// This function sort the elements in ascending order
```

```
void Quick sort (int A[], int p, int r)
{
    int x = A[r];
    int i, j;
    i = p-1;
    for (j = p; j <= r-1; j++)
    {
        if (A[j] <= x)
        {
            i = i+1
            exchange (&A[i], &A[j]);
        }
    }
    exchange (&A[i+1], &A[r]);
    return i+1;
}

// This function exchanges the given 2 values
   using call by references
void exchange (int* x, int* y)
{
    int temp;
    temp = *x;
*   x = *y;
*   y = temp;
}
```

* PROGRAM - 6 Rec Bin search. C

```
# include <stdio.h>
int Rec Bin search (int A[], int Data, int low, int high);
Void print Array (int A[], int n);

main()
{
    int [20], n, i, Data, index;
    Print f ("program for seaching a key in an
              array using Recursive binary search \n");
    Print f (" Author : Harsh Raj \n");
    print f ("Enter the no. of elements: ");
    Scan f ("% d", & n);
    Print f (" Enter the elements in ascending order: \n");
    for (i = 0; i <= n-1; i++)
    {
        Scan f ("% d ", & A [i]);
    }
    Print f ("\n Enter the element to be Searched:
    Scan f ("% d " & Data);
    Print f ("The given array is : [");
    Print Array (A, n);
    Print f ("] \n");
    Index = Rec BinSearch (A, Data, 0, n-1);
    if index >= 0)
    {
        Print f ("\n Data found at location % d", index
    }
else
```

```
        }
        Printf("Data not found \n");
    }
int RecBinsearch(int A[], int Data, int low, int high)
{
    int mid;
    While (low <= high)
    {
        mid = (low + high)/2;
        if (A[mid] == Data)
        {
            return mid;
        }
        else if (A[mid] < Data)
        {
            return RecBin search (A, Data, mid+1, high)
        }
        else
        {
            return RecBin search (A, Data, low, mid-1)
        }
    }Return
    Return -1;
}
// This function prints an Array
void printArray (int A[], int n)
{
    int i;
```

```
for (i=0; i<=n-1; i++)
{
    Printf ("%d", A[i]);
    Printf (" ");
}
```

* PROGRAM-7 Bin search.c

```
# include <stdio.h>
int Bin search (int A[], int data, int low, int high);
Void print Array (int A[], int n)
main ()
{
  int A [20], n, i, Data, index;
  Printf (" Program for searching a key in an array
          using non-Recusing binary search \n");
  printf (" Author: Harsh Raj ");
  Printf (" Experiment date: 23-02-2022");
  Printf (" Submission date: 27-03-2022 \n\n");

  Printf ("Enter the no. of elements: ");
  Scanf ("%d ", &n);
  Printf (" Enter array elements in ascending order:
  for (i=0; i<=n-1; i++)
  {
    Scanf ("%d", &[A]i);
  }

Printf ("\n Enter the element to be searched: ");
```

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

Printf ("\n The given array is : [ ");
Print Array (A, 0);
Printf ("]\n");

index = Bin search (A, Data, 0, n-1);
if (index >= 0)
{
Print f ("\n Data found at locator %d ", index);
}
else
{
Print f ("Data not found \n");
}
}

// function prints elements in an array.
void print Array (int A[], int n)
{
  int i;
  for (i = 0; i <= n-1; i++)
  {
    Print f ("%d ", A [i])
    Point f (" ");
  }
}

int Bin search (int A[], int Data, int low, int high)
{
```

```
int mid;
while (low <= high)
{
  mid = (low + high)(2);
  if (A[mid] == Data)
  {
   return mid;
  }
  else if (Data < A[mid]){
     high = mid - 1;
  }
  else {
     low = mid + 1;
  {
```

MODULE-3

1. Single Linked List.c

```
# include <stdio.h>
# include <std lib.h>
# <conio.h>

// creating structure for engle inked list struct
 slink list.
{
   ink data
   struct slink list * next
};

// renaming slink list as node
type def Struct slinklist node;

node * start = NULL;

int menu ()
{
int ch
Print f ("\n create a list");
Print f ("\n");
Print f ("\n2. Insert a node at begining \n");
Print f ("\n3. Insert a node at end");
Print f ("\n4. Insert a node at specified position");
Print f ("\n");
Print f ("\n 5. Delete a node at beginning");
print f ("\n6. Delete a node at last");
Print f ("\n7. Delete a node at specified position);
```

```
printf("\n8. Traverse the list (Left to Right);
Printf("\n9. Traverse the list (Right to left);
 Printf("\n");
 Printf("\n10. Count nodes");
 Printf("\n11. Exit \n");
 Printf("\n\n Enter your choice : ");
 Scanf("%d") & ch);
 return ch;
}

Function to get node dynamically
node * get node()
{
  node * newnode
  newnode = (node*) malloc (size of (node));
  printf("\n Enter data : ");
  scanf("%d", & new node → data);
  newnode → next = NULL;
  return new node;
}

// function to count the number of nodes in the
   linked list

int  countnode (node * ptr)
{
  int  count = 0;
  while (ptr != NULL)
  {
     count ++;
```

```
            ptr = ptr -> next;
        }
        return (count);
}

// function to create a single linked list with n nodes
void create list (int n)
{
    int i;
    node * newnode;
    node * temp;
    for (i=1; i<=n; i++)
    {
        newnode = getnode ();
        if (start == NULL)
        {
            start = new node;
        }
        else
        {
            temp = start;
            while (temp -> next != NULL)
            temp = temp -> next;
            temp -> next = new node;
        }
    }
}

// function to insert a node at the begining of the linked list
```

```
Void insert_at_beg()
{
  node * newnode
  newnode = getnode();
  if (start == NULL)
  {
    start = newnode
  }
  else
  {
    newnode -> next = start;
    Start = new node;
  }
  Printf("\n New Node has been inverted at the
          beginning ... \n");
}

// function to insert a node at the end of the linked
   list
Void insert_at_end()
{
  node * newnode, * temp;
  newnode = getnode();
  if (start == NULL)
  {
    start = newnode;
  }
  else
  {
    temp = start;
```

```
    while (temp -> next != NULL)
    temp = temp -> next;
    temp -> next = newnode;
    }
    printf ("\n New Node has been inserted at the
                end \n");
}

// function to insert a node at specified position in
   the linked list
void insert_at_mid()
{
  node * newnode, * temp, * prev;
  int pos, node ctr, ctr = 1;
  newnode = getnode();
  printf ("\n Enter the position : ");
  Scanf ("%d", & pos);
  nodectr = countnode (start);
  if (pos > 1 && pos < node ctr).
  {
     temp = prev = start;
     while (ctr < pos)
     {
        prev = temp;
        temp = temp -> next;
        ctr ++;
     }
  else
  printf (" position % d is not a middle
            position ", pos);
}
```

// function to delete a node from the beginning

```
void delete_at_beg ()
{
  node *temp;
  if (start == NULL)
  {
    Printf("\n No nodes are exist..");
    return;
    }
    else
    {
      temp = start;
      start = temp -> next
      free (temp);
      print ("\n Node deleted");
    }
}
```

// function to delete a node from specified position

```
void delete_at_mid ()
{
  Int ctr = 1, pos, node ctr;
  node * temp, * prev;
  if (start == NULL)
  {
    print f ("\n Empty List..");
    return;
  }
  else
  {
```

```
Print f ("\n Enter position of node to delete:");
Scan f ("%d", & pos);
node ctr = count node (start);
if (pos > node ctr
{
                    This node doesn't exist
print f(
if (pos >, 1 & pos < node ctr)
{
        temp = prev = start;
        while (ctr < pos)
        {
         prev = temp)
         temp = temp → next;
         ctr ++
        }
        prev → next = temp → next;
        free (temp);
        printf("\n Node deleted..");
    }
    else
    { print f ("\n Invalid position"); get ch();
    }
  }
}
// function to traverse a linked list from left to righ
void traverse ()
{  node * temp
   temp = start;
   print f ("\n Contents of list (left to Right): \n");
```

```
if (start == NULL)
{
    Printf ("\n Empty List");
    return;
}
else
{
  while (temp != NULL)
   {
     printf ("%d -->", temp -> data);
     temp = temp -> next;
   }
}
    Print ("X");
}
// Right to left
Void rev = traverse (node* start)
{
   if (start == NULL)
   {
      return;
   }
   else {
      rev- traverse (start -> next);
      printf ("%d -->", start -> data);
   }
}

main()
{
```

```
int ch,n
while (1){
    ch = menu ();
    switch (ch)
    {
        case 1:
            if (start == NULL) {
        Printf ("\n No. of nodes you want:");
        Scanf ("%d", &n);
        Create List (n);
        printf ("\n List created..\n");
    }
    else
    Printf (" List is already created \n");
break;
Case 2:
        insert_at_beg();
break;
Case 3:
        insert_at_mid;
Case 4:
        insert_at_mid;
Case 5:
        delete_at_beg();
        break;
Case 6:
        delete_at_last();
        break;
Case 7:
        delete_at_mid();
```

```
        break;
case 8: traverse ();
        break;
Case 9: printf(" contents of list (Right to left): \n");
        rev_traverse (start);
        break;
Case 10: print f ("\n No. of nodes : %d ", count-
                node (start);
```

DOUBLE LINKED LIST.C

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

Struct dlinklist
{
   Struct dlink list * left;
   int data;
   Struct dlinklist * right;
};

typedef Struct dlinklist node;
 node * start = NULL;

int menu () {
   int ch
   Print f (" \n 1. Create ");
   Print f ("\n ");
   print f ("\n 2. Insert a node at beginning");
   print f ("\n 3. Insert a node at end");
```

```
Print f ("\n4. Insert node at Specific position");
Print f ("\n5. Delete a node at beginning ");
Print f ("\n6. Delete a node from ~~middle~~ last ")
Print f ("\n7. Delete a node from middle ");
Print f ("\n8. Traverse the list from left to
                                    right ");
print f ("\n9. Traverse the list from right to left");
Print f ("\n10. Count no. of nodes in the list ")
printf ("\n 11. Exit ");
Print f ("\n\n Enter your choice : ");
Scanf ("%d", & ch );
return ch;
}
> node * get node ();
{
node * newnode;
newnode = (node*) malloc (size of (node));
Print f ("\n Enter data : ");
Scan f ("%d", & newnode → data);
newnode → left = NULL;
new node → right = NULL;
return newnode;
}
> void create list (int n)
{
int i;
node* newnode;
node* temp;
for (i = 1; i <= n; i++)
{
```

```
        newnode = getnode();
        if (start == NULL)
            start = newnode;
        else
        {
            temp = start;
            while (temp -> right;)
            temp -> right = newnode;
            newnode -> left = temp;
        }
    }
}
> int countnode (node * start)
{
    if (start == NULL)
        return 0;
    else
        return 1 + (countnode (start -> right));
}

> void all_insert_beg()
{
    node * newnode;
    newnode = getnode();
    if (start == NULL)
        start = newnode;
    else {
        newnode -> right = start;
        start -> left = newnode;
        start = newnode;
}
```

```
    printf("\n New node has been inserted \n");
}

> void all_insert_end()
{   node * newnode, * temp;
    newnode = getnode();
    if (start == NULL)
        start = new node;
    else {
        temp = start;
        while (temp -> right != NULL)
            temp = temp -> right;
            temp -> right = newnode
            newnode -> left = temp
    printf("New node inserted at the end");
}

> void all_insert_mid()
{
  node * newnode * temp;
  int pos, node ctr, ctr = 1;
  newnode = getnode();
  printf("\n Enter the position:");
  scanf("%d", & pos);
  node ctr = countnode(start);
  if (pos - node ctr >= 2)
  {
     printf("posisition is out of Range..");
     return;
  }
```

```
if (pos > 1 && pos < node.ctr) {
    temp = start;
    while (ctr < pos - 1)
    {
        temp = temp -> right;
        ctr++;
    }
    new node -> left = temp;
    newnode -> right = temp -> right;
    temp -> right -> left = newnode;
    temp -> right = newnode;
    }
    else (position isn't intermediate")
}
```

```
> void all_delete_beg()
{
    node * temp;
    if (start == NULL)
    {
        printf("\n Entry List");
        getch();
        return;
    }
    else {
        temp = start;
        start = start -> right;
        start -> left = NULL;
        free(temp);
        printf("\n node deleted");
    }
}
```

```
> void all_delete_end()
{
    node * newnode, * temp
    newnode = getnode();
    if (start == NULL)
        start = newnode;
    else {
        temp = start;
        while (temp -> weight;
        temp -> weight = newnode;
        newnode -> left = temp;
    }
    printf("\n New Node inserted at the end");
}

> void all_insert_mid()
{
    node * newnode * temp;
    int pos, node ctr, ctr=1
    newnode = getnode();
    Printf("\n Enter the position :");
    scanf("%d" & pos);
    node ctr = count node (start);
    if (pos - node ctr >= 2)
    {
        Printf("\n position is out of range");
        return;
    }
    if (pos > && pos < node ctr)
    {
```

```
temp = start;
while (ctr < pos - 1)
{
    temp = start; temp -> right;
    while ctr++
}
    new node -> left = temp;
    new node -> right = temp -> right;
    temp -> right -> left = newnode;
    temp -> right = newnodes;
    printf("new node inserted at position");
}
else
    Print f ("position %d of list is not
    an intermediate position");

> Void all_delete_beg()
{
  node* temp;
  if (start == NULL)
    {
      Print f ("\n Empty list");
      get ch();
      return
    }
    else {
        temp = start;
        start = start -> right;
        start -> left = NULL;
        free (temp);
```

```
        printf ("\n NODE deleted");
    }
}

> Void all_delete_last()
{
  node *temp
    if (start == NULL){
      printf("\n Empty List");
      getch();
      return;
    }
    else {
        temp = start;
        While(temp -> right != NULL)
            temp = temp -> right
            temp -> left -> right = NULL;
              free (temp);
              temp = (NULL);

> void all_delete_mid()
{
   int i = 0, pos, node ctr;
   node * temp;
   if (start == NULL)
   printf("\n Empty List");
   getch();
   return;
}
   else
{
```

```
Print f ("\n Enter the position of node to del")
Scan f ("%d", & pos);
node ctr = Count node (start);
if (pos > node ctr) {
    print f ("This node doesn't exist");
    get ch ();
    return
}
else if (pos > 1 && pos < node ctr)
{
    temp = start;    i = 1;
    while (i < pos)
    { temp = temp → right;
      i++
    }
    temp = right → left = temp → left;
    temp → left → right = temp → right;
    free (temp);
    Print f (node deleted");
  else {
    Print f (invalid position)
    get ch();
  }
}

} void traverse_left_to_right () {
    node * temp;
    temp = start;
    Print f ("The Contents of List (left to Right)
    if (start == NULL)
```

```
        Print f ("\n Empty List");
        else {
            While (temp != NULL)
            {
              Print f (%d -> "temp -> data;)
               temp = temp -> right;
            }
        }
        Print f ("X");
    }

> Void traverse right-to-left ()
  {
    node * temp;
    temp = start;
    print f ("\n The contents of list (Right to left)");
     if (start == NULL)
    Print f ("\n Empty List");
    else
    {
          While (temp -> right != NULL)
            temp = temp -> right;
    }
    While (temp != NULL){
      {
          Print f (%d --> ", temp -> data);
           temp = temp -> left;
        }
  }
```

```
main (){
int ch, n
Printf (" C program to illustrate the operative on
          double linked list ")
printf (" Author= Harsh Raj \n");
print f ( Experiment date:
 print f ( Submission date: 27-03-2022 \n");
 print f ("\n");
 while (1){
      ch= menu ();
       Switch (ch)
       {
        case 1:
        Print f ( Enter " no. of nodes to create: ");
        Scan f ("%d", &n);
        CreateList (n);
        break;
        case 2: all_insert_at_beg ();
        break;
        case 3: all_insert_at_end ();
        break;
        Case 4; all_insert_at_mid();
        break;
        case 5; all_delete_at_beg ();
        break;
        case 6; all delete_at_end ();
        break;
        case 7; all delete_at_mid ();
        break;
        case 8: traverse_left_to_right();
```

```
        break;
        Case 9: traverse_right_to_left();
         break;
         exit (0);
    }
```

Circular Single Linked List.c

```
# include <stdio.h>
# include <conio.h>
# include <stdlib.h>
  struct cslinklist
{
    int data
    Struct cslinklist * next;
}
typedef struct cslinklist node;
node * start = NULL;

int nodectr;
node * get code ()
{
   node * newnode
   newnode = (node*) malloc (size of (node));
   Print f (" \n Enter data : ");
    Scan f ("%d, & newnode -> data);
   newnode -> next = NULL; return;
   newnode;
}
```

```
int menu(){
 int ch;
 clrscr();
 printf("\n1. Create a List");
 printf("\n2. Insert node at beginning");
 printf("\n3. Insert node at end");
 printf("\n");
 printf("\n4. Delete node at beginning");
 printf("\n5. Delete node at last");
 printf("\n7. Delete node at middle");
 printf("\n");
 printf("\n8. Display the list");
 printf("\n9. Exit");
 printf("\n10. Enter your choice");
 scanf("%d", &ch);
 return ch;
}

> void creatlist(int n)
{
 int i;
 node * newnode;
 node * temp;
 node ctr = n;
 for(i=0; i<n-i++){
     newnode = getnode();
     if(start == NULL){
         start = newnode;
     }
     else {
```

```
        temp = start
        while (temp -> next != NULL)
          temp = temp -> next;
          temp -> next = new node;
      }
    }
    new node -> next = start;
}

void display (){
  node * temp ;
  printf ("\n the contents of list (L to R)");

  if (start == NULL)
  printf ("\n Empty list");
  else {
    do {
        printf ("\t %d", temp -> data);
        temp -> temp -> next;
    } while (temp != start);
    printf ("X");

  }
}

void all_insert_beg ()
{
  node * newnode, * last ;
  new node = getnode ()
  if (start == NULL ) {
```

```
        start = newnode;
        newnode -> next = start;
    }
    else {
        last = start;
        while (last -> next != start)
        last = last -> next;
        newnode -> next = start;
        start = newnode;
        last -> next = start;
    }
    else {
        last = start;
        while (last -> next != start)
        last = last -> next;
        newnode -> next = start;
        start = newnode;
        last -> next = start;
    }
    Bint f ("\n node inserted at bginning");
    node ctr++;
}

> void all_insert_end ()
    node* newnode, * temp;
    newnode = getnode ()
    if (start == NULL){
        start = newnode;
        newnode -> next = start;
    }
```

```
else {
    temp = start;
    while (temp -> next != start)
     temp = temp -> next;
    temp -> next = newnode;
    newnode -> next = start;
  }
  Print f ("\n Node inserted at end..");
   node ctr++
}

> void all_insert_mid () {
  node * newnode, * temp, * prev;
  int i, pos;
  newnode = getnode();
  Print f ("\n Enter the position:");
  Scan f ("%d", & Pos
  if (Pos > 1 && pos < node ctr) {
      temp = start;
      Prev = temp;
      i = 1;
      while (i < pos) {
       prev = temp
        temp = temp -> next; i++;
    }
      Prev -> next = newnode
      newnode -> next = temp;
      node ctr ++
      Print f ("\n Node Inserted at middle");
}
```