LAB MANUAL

Data Structures

B.TECH I YEAR- II SEMESTER

(R23)

DEPARTMENT OF Computer Science Engineering

Mother Theresa Institute of Engineering &Technology

Melumoi Village And Post, Palamaner,, Chittoor Dist, Andhra Pradesh 517408

(Approved by AICTE. New Delhi Affiliated to JNTUA Ananthapuramu.

(23A05201P) Data Structures Lab
(Common to All Branches of Engineering)

Course Objectives:

1. To introduce to the different data structures
2. To elucidate how the data structure selection influences the algorithm complexity
3. To explain the different operations that can be performed on different data structures
4. To introduce to the different search and sorting algorithms.

Laboratory Experiments

1. String operations using array of pointers
2. Searching Algorithms (With the Number of Key Comparisons) Sequential, Binary and Fibonacci Search Algorithms.
3. Sorting Algorithms: Insertion Sort, Selection Sort, Shell Sort, Bubble Sort, Quick Sort, Heap Sort, Merge Sort, and Radix Sort. Using the system clock, compute the time taken for sorting of elements. The time for other operations like I/O etc should not be considered while computing time.
4. Implementation of Singly Linked List, Doubly Linked List, Circular Linked List
5. Stack implementation using arrays
6. Stack implementation using linked lists
7. Queue implementation using arrays. Implement different forms of queue. While implementing you should be able to store elements equal to the size of the queue. No positions should be left blank.
8. Queue implementation using linked lists
9. Creation of binary search tree, performing operations insertion, deletion, and traversal.
10. Breadth first search
11. Depth first search
12. Travelling sales man problem
13. File operations
14. Indexing of a file
15. Reversing the links (not just displaying) of a linked list.
16. Consider a linked list consisting of name of a person and gender as a node. Arrange the linked list using ‘Ladies first’ principle. You may create new linked lists if necessary.
17. An expression can be represented in three ways: infix, prefix and postfix. All the forms are necessary in different contexts. Write modules to convert from one form to another form.
18. A table can be defined as a collection of rows and columns. Each row and column may have a label. Different values are stored in the cells of the table. The values can be of different data types. Numerical operations like summation, average etc can be performed on rows/columns which contain numerical data. Such operations are to be prevented on data which is not numeric. User may like to insert row/columns in the already existing table.

User may like to remove row/column. Create table datatype and support different operations on it.

Course Outcomes:

At the end of the course students should be able to

1. Select the data structure appropriate for solving the problem (L5)
2. Implement searching and sorting algorithms (L3)
3. Design new data types (L6)
4. Illustrate the working of stack and queue (L4)
5. Organize the data in the form of files (L6)

CO1: Explain the role of linear data structures in organizing and accessing data efficiently in algorithms.
CO2: Design, implement, and apply linked lists for dynamic data storage, demonstrating understanding of memory allocation.
CO3: Develop programs using stacks to handle recursive algorithms, manage program states, and solve related problems.
CO4: Apply queue based algorithms for efficient task scheduling and breadth first traversal in graphs and distinguish between deques and priority queues and apply them appropriately to solve data management challenges.
CO5: Recognize scenarios where hashing is advantageous, and design hash based solutions for specific problems.

Sl.no	List of Experiments
1	String operations using array of pointers
2	Searching Algorithms (With the Number of Key Comparisons) Sequential, Binary and Fibonacci Search Algorithms.
3	Sorting Algorithms: Insertion Sort, Selection Sort, Shell Sort, Bubble Sort, Quick Sort, Heap Sort, Merge Sort, and Radix Sort. Using the system clock, compute the time taken for sorting of elements. The time for other operations like I/O etc should not be considered while computing time.
4	Implementation of Singly Linked List, Doubly Linked List, Circular Linked List
5	Stack implementation using arrays
6	Stack implementation using linked lists
7	Queue implementation using arrays. Implement different forms of queue. While implementing you should be able to store elements equal to the size of the queue. No positions should be left blank.
8	Queue implementation using linked lists
9	Creation of binary search tree, performing operations insertion, deletion, and traversal.
10	Breadth first search
11	Depth first search
12	Travelling sales man problem
13	File operations
14	Indexing of a file
15	Reversing the links (not just displaying) of a linked list.
16	Consider a linked list consisting of name of a person and gender as a node. Arrange the linked list using ‘Ladies first’ principle. You may create new linked lists if necessary.
17	An expression can be represented in three ways: infix, prefix and postfix. All the forms are necessary in different contexts. Write modules to convert from one form to another form.
18	A table can be defined as a collection of rows and columns. Each row and column may have a label. Different values are stored in the cells of the table. The values can be of different data types. Numerical operations like summation, average etc can be performed on rows/columns which contain numerical data. Such operations are to be prevented on data which is not numeric. User may like to insert row/columns in the already existing table. User may like to remove row/column. Create table datatype and support different operations on it.

1. Write a C program that sorts the strings using array of pointers

Aim: Design a C program that sorts the strings using array of pointers

Source code:-

```
#include<stdio.h>
#include <stdlib.h>
#include <string.h>
void main()
{
char * temp;
int i, j, diff, num_strings;
char * strArray[10];
clrscr();
printf("Enter the number of strings: ");
scanf("%d",&num_strings);
for (i = 0; i < num_strings ;i++)
 {
strArray[i] = (char *)malloc(sizeof(char)*20);
printf("Enter string %d: ", i+1);
scanf("%s",strArray[i]);
}
printf("Before Sorting\n");
for (i = 0; i < num_strings ;i++)
{
printf("%s\n",strArray[i]);
}
for (i = 0; i < num_strings - 1; i++)
{
for (j = 0; j < num_strings-1-i; j++)
{
diff = strcmp(strArray[j], strArray[j+1]);
if (diff > 0)
 {
temp = strArray[j];
strArray[j] = strArray[j+1];
strArray[j+1] = temp;
}
}
}
printf("After Sorting\n");
for (i = 0; i < num_strings ;i++)
{
printf("%s\n",strArray[i]);
}
getch();
```

```
}
```

INPUT:-

OUTPUT:-

RESULT:-

2. A)Write a program to search a key element within the given array of elements using linear search process

Aim: To write a program to search a key element within the given array of elements using linear search process

Source code:-

```
#include<conio.h>
#include<stdio.h>
void main()
{
int a[20], i, n, key, flag = 0, pos;
clrscr();
printf("Enter value of n : ");
scanf("%d", &n);
for (i = 0; i < n; i++)// this for loop is to enter n number of elements into the array
{
printf("Enter element for a[%d] : ", i);
scanf("%d", &a[i]);
}
printf("Enter key element : ");
scanf("%d", &key);
for (i = 0; i < n; i++) //this for loop is to search the elements from 0 index
 {
if (a[i] == key)   This if condition compares the each element with the key element
{
flag = 1;
pos = i;
}
}
if (flag == 1 )
{
printf("The key element %d is found at the position %d\n", key, pos);
}  printf("\n the search is successful");
else
 {
printf("The Key element %d is not found in the array\n", key);
}    printf("\n the search is unsuccessful");
getch();
}
```

INPUT:-

OUTPUT:-

RESULT:-

2.B) Write a program to search a key element in the given array of elements using binary search .

AIM:- Write a program to search a key element in the given array of elements using binary search .

Sourcecode:-

```
#include<conio.h>
#include<stdio.h>
void main()
 {
int a[20], i, j, n, key, flag = 0, low, high, mid, temp;
clrscr();
printf("Enter value of n : ");
scanf("%d", &n);
for (i = 0; i < n; i++)
{
printf("Enter element for a[%d] : ", i);
scanf("%d", &a[i]);
}
printf("Enter key element : ");
scanf("%d", &key);
for (i = 0; i < n - 1; i++) //This for loop is for to sort the elements
 {
for (j = 0; j < n - i - 1; j++)
 {
if (a[j] > a[j + 1]) //This if condition swaps the maximum value with min value
 {
temp =  a[j];
a[j] = a[j + 1];
a[j + 1] = temp;
}
}
}
printf("After sorting the elements in the array are\n");
for (i = 0; i < n; i++)
 {
printf("Value of a[%d] = %d\n", i, a[i]);
}
low = 0;
high = n - 1;
while (flag == 0 && low <= high)
{
mid = (low + high) / 2;
if (a[mid] == key)
 {
flag = 1;
break;
```

```
}
 else if (a[mid] < key)
  {
low = mid + 1;
}
else if (a[mid] > key)
{
high = mid - 1;
}
}
if (flag == 1)
{
printf("The key element %d is found at the position %d\n", key, mid);
}   printf("\n the search is successful");
else
{   printf("\n the search is unsuccessful");
printf("The Key element %d is not found in the array\n", key);
}
getch();
}
```

INPUT:-

OUTPUT:-

RESULT:-

2.c) Write a C program to implement Fibonacci search technique

AIM:- To write C program to implement Fibonacci search technique

Source Code:

```
#include  <stdio.h>
#include <conio.h>
int main()
 {
int size;
int *arr, i,x,result=-1;
clrscr();
printf("Enter the size of an array: ");
scanf("%d",&size);
arr = (int*) malloc(size * sizeof(int));
printf("Enter the %d array elements\n",size);
for (i = 0; i < size; i++)
 {
scanf("%d", &arr[i]);
}
printf("Enter the element to be searched: ");
scanf("%d",&x);
result = fibonaccianSearch(arr,x,size+1);
if (result != -1 )
printf("Element found at index: %d.\n",result);
else
printf("Element not found.\n");
getch();
return 0;
}
int min(int x,int y)
 {
return (x<=y)? x:y;
}
int fibonaccianSearch(int arr[],int x,int n)
 {
int offset=-1;
int m2=0;
int m1=1;
int m=m2+m1;
while(m<n)
 {
m2=m1;
m1=m;
m=m2+m1;
}
```

```
while(m>1)
 {
int i=min(offset+m2,n-1);
if (arr[i]<x)
 {
m=m1;
m1=m2;
m2=m-m1;
offset=i;
}
else if(arr[i]>x)
{
m=m2;
m1=m1-m2;
m2=m-m1;
}
else
return i;
}
if (m1 && arr[offset+1]==x)
return offset+1;
return -1;
}
```

INPUT:-

OUTPUT:-

RESULT:-

3. Sorting Algorithms: Insertion Sort, Selection Sort, Shell Sort, Bubble Sort, Quick Sort, Heap Sort, Merge Sort, and Radix Sort. Using the system clock, compute the time taken for sorting of elements. The time for other operations like I/O etc should not be considered while computing time.

3.A) Write a c program to implement Insertion Sort

AIM:-To write a c program to implement insertion sort

Source code:-

```
#include<conio.h>
#include<stdio.h>
void main()
 {
int a[20], i, n, j, temp;
clrscr();
printf("Enter value of n : ");
scanf("%d", &n);
for (i = 0; i < n; i++)
 {
printf("Enter element for a[%d] : ", i);
scanf("%d", &a[i]);
}
printf("Before sorting the elements in the array are\n");
for (i = 0; i < n; i++)
 {
printf("Value of a[%d] = %d\n", i, a[i]);
}
for (i = 1; i < n; i++)
 {
temp = a[i];
for (j = i; j > 0; j--)
 {
if (a[j - 1] > temp)
 {
a[j] = a[j - 1];
a[j - 1] = temp;
}
}
}
printf("After sorting the elements in the array are\n");
for (i = 0; i < n; i++) {
printf("Value of a[%d] = %d\n", i, a[i]);
}
getch();
}
```

INPUT:-

OUTPUT:-

RESULT:-

3.B) Write a c program to implement Selection Sort

AIM:-To write a c program to implement selection sort

Source code:-

```
#include<conio.h>
#include<stdio.h>
void main()
{
int a[20], i, n, j, small, index;
clrscr();
printf("Enter value of n : ");
scanf("%d", &n);
for (i = 0; i < n; i++)
 {
printf("Enter element for a[%d] : ", i);
scanf("%d", &a[i]);
}
printf("Before sorting the elements in the array are\n");
for (i = 0; i < n; i++)
 {
printf("Value of a[%d] = %d\n", i, a[i]);
}
for (i = 0; i < n; i++)
{
small = a[i];
index = i;
for (j = i + 1; j < n; j++)
 {
if (a[j] < small)
{
small = a[j];
index = j;
}
}
a[index] = a[i];
a[i] = small;
}
printf("After sorting the elements in the array are\n");
for (i = 0; i < n; i++)
 {
printf("Value of a[%d] = %d\n", i, a[i]);
}
getch();
}
```

INPUT:-

OUTPUT:-

RESULT:-

3.C) Write a c program to implement Shell Sort

AIM:-To write a c program to implement Shell sort

Source code:-

```
#include <stdio.h>
#include <conio.h>
void printArray(int arr[],int);
int main()
{
int size;
int *arr, i;
clrscr();
printf("Enter array size : ");
scanf("%d",&size);
arr = (int*) malloc(size * sizeof(int));
printf("Enter %d elements : ",size);
for (i = 0; i < size; i++)
 {
scanf("%d", &arr[i]);
}
printf("Before sorting the elements are : ");
printArray(arr,size);
shellSort(arr,size);
printf("After sorting the elements are : ");
printArray(int arr[],int n);
getch();
return 0;
}
int shellSort(int arr[], int n)
{
int gap,i;
for (gap = n/2; gap > 0; gap /= 2)
 {
for (i = gap; i < n; i += 1)
 {
int temp = arr[i];
int j;
for (j = i; j >= gap && arr[j - gap] > temp; j -= gap)
arr[j] = arr[j - gap];
arr[j] = temp;
}
}
return 0;
}
```

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

INPUT:-

OUTPUT:-

RESULT:-

3.D) Write a c program to implement Bubble Sort

AIM:-To write a c program to implement Bubble sort

Source code:-

```
#include<stdlib.h>
#include <stdio.h>
#include <conio.h>
void printArray(int arr[],int);   int bubblesort(int [], int);
int main()
{
int size;
int *arr, i;
clrscr();
printf("Enter array size : ");
scanf("%d",&size);
arr = (int*) malloc(size * sizeof(int));
printf("Enter %d elements : ",size);
for (i = 0; i < size; i++)
 {
scanf("%d", &arr[i]);
}
printf("Before sorting the elements are : ");
printArray(arr,size);
bubblesort shellSort(arr,size);
printf("After sorting the elements are : ");
printArray(int arr[],int n);
getch();
return 0;
}
 bubblesort
int shellSort(int arr[], int n)
{
int gap,i;
for (gap = n/2; gap > 0; gap /= 2)
 {
for (i = gap; i < n; i += 1)
 {
int temp = arr[i];
int j;
for (j = i; j >= gap && arr[j - gap] > temp; j -= gap)
arr[j] = arr[j - gap];
arr[j] = temp;
}
}
return 0;
}
```

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

INPUT:-

OUTPUT:-

RESULT:-

3.E) Write a c program to implement Quick sort Sort

AIM:-To write a c program to implement quick sort

Source code:-

```
#include <stdio.h>
void main()
{
int arr[15], i, n;
printf("Enter array size : ");
scanf("%d", &n);
printf("Enter %d elements : ", n);
for (i = 0; i < n; i++)
 {
scanf("%d", &arr[i]);
}
printf("Before sorting the elements are : ");
display(arr, n);
quickSort(arr, 0, n - 1);
printf("After sorting the elements are : ");
display(arr, n);
}
void display(int arr[15], int n)
{
int i;
for (i=0;i<n;i++)
printf("%d ", arr[i]);
printf("\n");
}
void quickSort(int arr[15], int low, int high)
{
int j;
if (low<high)
{
j = partition(arr, low, high);
quickSort(arr,low,j-1);
quickSort(arr,j+1,high);
}
}
int partition(int arr[15], int lb, int ub)
{
int pivot, down = lb, up = ub, temp;
pivot = arr[lb];
while (down<up)
{
while (arr[down]<=pivot && down<up)
```

```
{
down++;
}
while (arr[up]>pivot)
{
ented by 1
up--;
}
if (down<up)
{
temp = arr[down];
arr[down] = arr[up];
arr[up] = temp;
}
}
arr[lb] = arr[up];
arr[up] = pivot;
return up;
}
```

INPUT:-

OUTPUT:-

RESULT:-

3.F) Write a program to sort (ascending order) the given elements using heap sort technique.

AIM:- Write a program to `sort ( ascending order )` the given elements using `heap sort` technique.

Source Code:-

```
#include <stdio.h>
void main()
 {
int arr[15], i, n;
printf("Enter array size : ");
scanf("%d", &n);
printf("Enter %d elements : ", n);
for (i = 0; i < n; i++)
{
scanf("%d", &arr[i]);
}
printf("Before sorting the elements are : ");
display(arr, n);
heapsort(arr,n);
printf("After sorting the elements are : ");
display(arr, n);
}
void display(int arr[15],int n)
{
int i;
for (i=0;i<n;i++)
{
printf("%d ",arr[i]);
}
printf("\n");
}
void heapify(int arr[],int n,int i)
{
int largest=i;
int l=2*i+1;
int r=2*i+2;
int temp;
if (l<n && arr[l]>arr[largest])
largest=l;
if(r<n && arr[r]>arr[largest])
largest=r;
if (largest!=i)
 {
temp=arr[i];
arr[i]=arr[largest];
arr[largest]=temp;
```

```
heapify(arr,n,largest);
}
}
void heapsort(int arr[],int n)
 {
int i,temp;
for (i=n/2-1;i>=0;i--)
{
heapify(arr,n,i);
}
for (i=n-1;i>=0;i--)
{
temp=arr[0];
arr[0]=arr[i];
arr[i]=temp;
heapify(arr,i,0);
}
}
```

INPUT:-

OUTPUT:-

RESULT:-

3.G) Write a C program to Sort given elements using Merge sort

AIM:- Write a program to sort (Ascending order) the given elements using merge sort technique.

Source Code:-

```
#include <stdio.h>
void main()
{
int arr[15], i, n;
printf("Enter array size : ");
scanf("%d", &n);
printf("Enter %d elements : ", n);
for (i = 0; i < n; i++) {
scanf("%d", &arr[i]);
}
printf("Before sorting the elements are : ");
display(arr, n);
splitAndMerge(arr, 0, n - 1);
printf("After sorting the elements are : ");
display(arr, n);
}
void display(int arr[15], int n)
{
int i;
for(i=0;i<n;i++)
{
printf("%d ", arr[i]);
}
printf("\n");
}
void merge(int arr[15], int low, int mid, int high)
 {
int i=low,h=low,j=mid+1,temp[15];
while(i<=mid && j<=high)
{
if(arr[i]<arr[j])
 {
temp[h]=arr[i];
i++;
h++;
}
else
{
temp[h]=arr[j];
j++;
```

```
h++;
}
}
while(i<=mid)
temp[h++]=arr[i++];
while(j<=high)
temp[h++]=arr[j++];
for(i=low;i<h;i++)
{
arr[i]=temp[i];
}
}
void splitAndMerge(int arr[15], int low, int high)
 {
if(low<high) {
int mid=(low+high)/2;
splitAndMerge(arr,low,mid);
splitAndMerge(arr,mid+1,high);
merge(arr,low,mid,high);
}
}
```

INPUT:-

OUTPUT:-

RESULT:-

3.H) Write a c program to implement Radix Sort

AIM:-To write a c program to implement Radix sort

Source code:-

```
#include <stdio.h>
#include <conio.h>
void printArray(int a[],int);
int main()
{
int size;
int *arr, i;
printf("Enter array size : ");
scanf("%d",&size);
arr = (int*) malloc(size * sizeof(int));
printf("Enter %d elements : ",size);
for (i = 0; i < size; i++) {
scanf("%d", &arr[i]);
}
printf("Before sorting the elements are : ");
printArray(arr,size);
RadixSort(arr,size);
printf("After sorting the elements are : ");
return 0;
}
int largest(int a[], int n)
 {
int i,largest=a[0];
for (i=0;i<n;i++)
 {
if (a[i]>largest)
largest=a[i];
}
return largest;
}
void printArray(int a[], int n)
 {
 int i;
for (i=0; i<n; i++)
printf("%d ",a[i]);
printf("\n");
}
void RadixSort(int a[], int n)
{
int i=0;
```

```
int bucket[10][10];
int bucket_count[10];
int j,k,remainder,NOP=0,divisor=1,large,pass;
large=largest(a,n);
while(large>0)
 {
NOP++;
large=large/10;
}
for (pass=0;pass<NOP;pass++)
 {
for (i=0;i<10;i++)
 {
bucket_count[i]=0;
}
for (i=0;i<n;i++)
 {
remainder=a[i]/divisor%10;
bucket[remainder][bucket_count[remainder]]=a[i];
bucket_count[remainder]++;
}
for (k=0;k<10;k++)
 {
for (j=0;j<bucket_count[k];j++)
 {
a[i]=bucket[k][j];
i++;
}
}
divisor=divisor*10;
}
}
```

INPUT:-

OUTPUT:-

RESULT:-

4 Implementation of Singly Linked List, Doubly Linked List, Circular Linked List

4. A) Write a c program to implement all single linked list operations

AIM:-Write a program that uses functions to perform the following operations on singly linked list i)Creation ii)insertion iii)deletion iv) Traversal

source code:-

```
#include <stdio.h>
#include <malloc.h>
#include <stdlib.h>
struct node
{
int value;
struct node *next;
};
void insert();
void display();
void delete();
int count();
typedef struct node DATA_NODE;
DATA_NODE *head_node, *first_node, *temp_node = 0, *prev_node, next_node;
int data;
int main()
{
int option = 0;
printf("Singly Linked List Example - All Operations\n");
while (option < 5)
{
printf("Options\n");
printf("1 : Insert into Linked List \n");
printf("2 : Delete from Linked List \n");
printf("3 : Display Linked List\n");
printf("4 : Count Linked List\n");
printf("5 : Exit()\n");
printf("Enter your option:");
scanf("%d", &option);
switch (option)
{
case 1:
insert();
break;
case 2:
delete();
break;
case 3:
```

```
display();
break;
case 4:
count();
break;
default:
break;
}
}
return 0;
}
void insert()
{
printf("Enter Element for Insert Linked List : ");
scanf("%d", &data);
temp_node = (DATA_NODE *) malloc(sizeof (DATA_NODE));
temp_node->value = data;
if (first_node == 0)
{
first_node = temp_node;
}
else
{
head_node->next = temp_node;
}
temp_node->next = 0;
head_node = temp_node;
fflush(stdin);
}
void delete()
 {
int countvalue, pos, i = 0;
countvalue = count();
temp_node = first_node;
printf("Enter Position for Delete Element : ");
scanf("%d", &pos);
if (pos> 0 &&pos<= countvalue)
{
if (pos == 1)
 {
temp_node = temp_node -> next;
first_node = temp_node;
printf("Deleted Successfully.");
}
```

```
else
 {
while (temp_node != 0)
{
if (i == (pos - 1))
{
prev_node->next = temp_node->next;
if(i == (countvalue - 1))
{
head_node = prev_node;
}
printf("Deleted Successfully.");
break;
}
else
{
i++;
prev_node = temp_node;
temp_node = temp_node -> next;
}
}
}
}
else
printf("Invalid Position.");
}
void display()
{
int count = 0;
temp_node = first_node;
printf("Display Linked List : \n");
while (temp_node != 0)
{
printf("# %d # ", temp_node->value);
count++;
temp_node = temp_node -> next;
}
printf("\nNo Of Items In Linked List : %d\n", count);
}
int count()
{
int count = 0;
temp_node = first_node;
while (temp_node != 0)
```

```
{
count++;
temp_node = temp_node -> next;
}
printf("No Of Items In Linked List : %d\n", count);
return count;
}
```

INPUT:-

OUTPUT:-

RESULT:-

4.B) Write a c program to implement All operations on double linked list.

Aim:- Write a program that uses functions to perform the following operations on double linked list

i) Creation ii)insertion iii)deletion iv) Traversal

Source Code:-

```
#include  <stdio.h>
#include <stdlib.h>
#include <conio.h>
struct dnode
{
struct dnode *prev;
int data;
struct dnode *next;
};
struct dnode *start = NULL;
void insert(int);
void remov(int);
void display();
int main()
{
int n, ch;
do
{
printf("Operations on doubly linked list");
printf("\n1. Insert \n2.Remove\n3. Display\n0. Exit");
printf("\nEnter Choice 0-4? : ");
scanf("%d", &ch);
switch (ch)
{
case 1:
printf("Enter number: ");
scanf("%d", &n);
insert(n);
break;
case 2:
printf("Enter number to delete: ");
scanf("%d", &n);
remov(n);
break;
case 3:
display();
```

```
break;
}
}while (ch != 0);
}
void insert(int num)
{
struct dnode *nptr, *temp = start;
nptr = malloc(sizeof(struct dnode));
nptr->data = num;
nptr->next = NULL;
nptr->prev = NULL;
if (start == NULL)
{
start = nptr;
}
else
{
while (temp->next != NULL)
temp = temp->next;
nptr->prev = temp;
temp->next = nptr;
}
}
void remov(int num)
{
struct dnode *temp = start;
while (temp != NULL)
{
if (temp->data == num)
{
if (temp == start)
{
start = start->next;
start->prev = NULL;
}
else
{
if (temp->next == NULL)
temp->prev->next = NULL;
else
{
temp->prev->next = temp->next;
temp->next->prev = temp->prev;
}
```

```
free(temp);
}
return ;
}
temp = temp->next;
}
printf("%d not found.\n", num);
}
void display()
{
struct dnode *temp = start;
while (temp != NULL)
{
printf("%d\t", temp->data);
temp = temp->next;
}
printf("\n");
}
```

INPUT:-

OUTPUT:-

RESULT:-

4.C) Write a c program to Implementation of Circular Queue using Dynamic Array

AIM:- Write a program to implement circular queue using dynamic array.

Source code:-

```
#include  <stdio.h>
#include <stdlib.h>
int main()
{
int op, x;
printf("Enter the maximum size of the circular queue : ");
scanf("%d", &maxSize);
initCircularQueue();
while(1)
{
printf("1.Enqueue 2.Dequeue 3.Display 4.Exit\n");
printf("Enter your option : ");
scanf("%d",&op);
switch(op) {
case 1:
printf("Enter element : ");
scanf("%d",&x);
enqueue(x);
break;
case 2:
dequeue();
break;
case 3:
display();
break;
case 4:
exit(0);
}
}
}
int *cqueue;
int front, rear;
int maxSize;
void initCircularQueue()
{
cqueue = (int *)malloc(maxSize * sizeof(int));
front = -1;
rear = -1;
}
void dequeue()
```

```
 {
if (front == -1)
 {
printf("Circular queue is underflow.\n");
}
else
{
printf("Deleted element = %d\n", *(cqueue + front));
if (rear == front)
{
rear = front = -1;
}
else if (front == maxSize - 1)
 {
front = 0;
} else {
front++;
}
}
}
void enqueue(int x) {
if (((rear == maxSize - 1) && (front == 0)) || (rear + 1 == front)) {
printf("Circular queue is overflow.\n");
} else {
if (rear == maxSize - 1) {
rear = -1;
} else if (front == -1) {
front = 0;
}
rear++;
cqueue[rear] = x;
printf("Successfully inserted.\n");
}
}
void display()
{
int i;
if (front == -1 && rear == -1)
{
printf("Circular queue is empty.\n");
}
else
{
printf("Elements in the circular queue : ");
```

```
if (front <= rear) {
for (i = front; i <= rear; i++)
{
printf("%d ", *(cqueue + i));
}
}
else
{
for (i = front; i <= maxSize - 1; i++)
{
printf("%d ", *(cqueue + i));
}
for (i = 0; i <= rear; i++)
 {
printf("%d ", *(cqueue + i));
}
}
printf("\n");
}
}
```

INPUT:-

OUTPUT:-

RESULT:-

5. Write a C program to implement different Operations on Stack using Array representation.

Aim: Write a program to implement stack using arrays.

Source Code:

```
#include <stdio.h>
#include <stdlib.h>
#define STACK_MAX_SIZE 10
#include "StackOperations.c"
int main() {
int op, x;
while(1) {
printf("1.Push 2.Pop 3.Display 4.Is Empty 5.Peek 6.Exit\n");

printf("Enter your option : ");
scanf("%d", &op);
switch(op) {
case 1:
printf("Enter element : ");
scanf("%d", &x);
push(x);
break;
case 2:
pop();
break;
case 3:
display();
break;
case 4:
isEmpty();
break;
case 5:
peek();
break;
case 6:
exit(0);
}
}
}
#include<stdio.h>
#include<stdlib.h>
#define STACK_MAX_SIZE 10
int arr[STACK_MAX_SIZE];
```

```
int top=-1;
int x;
void push(int element)
{
if (top==STACK_MAX_SIZE - 1)
 {
printf("Stack is overflow.\n");
} else {
top=top+1;
arr[top]=element;
printf("Successfully pushed.\n");
}
}
void pop()
{
if (top<0)
 {
printf("Stack is underflow.\n");
}
else
{
x=arr[top];
top=top-1;
printf("Popped value = %d\n", x);
}
}
void display()
{
if (top<0)
 {
printf("Stack is empty.\n");
} else {
printf("Elements of the stack are : ");
for(int i=top;i >=0;i--)
 {
printf("%d ", arr[i]);
}
printf("\n");
}
} void isEmpty()
{
if (top<0)
 {
printf("Stack is empty.\n");
```

```
}
else
 {
printf("Stack is not empty.\n");
}
}
void peek()
 {
if (top<0) {
printf("Stack is underflow.\n");
} else {
x=arr[top];
printf("Peek value = %d\n", x);
}
}
```

INPUT:-

OUTPUT:-

RESULT:-

6.Write a C program to implement different Operations on Stack using Linked Lists

Aim: Write a program to implement stack using linked lists.

Source Code:

```
#include  <stdio.h>
#include <stdlib.h>
int main()
{
int op, x;
while(1)
{
printf("1.Push 2.Pop 3.Display 4.Is Empty 5.Peek 6.Exit\n");
printf("Enter your option : ");
scanf("%d", &op);
switch(op)
{
case 1:
printf("Enter element : ");
scanf("%d", &x);
push(x);
break;
case 2:
pop();
break;
case 3:
display();
break;
case 4:
isEmpty();
break;
case 5:
peek();
break;
case 6:
exit(0);
}
}
}
#include  <stdio.h>
#include <stdlib.h>
struct stack
{
```

```
int data;
struct stack *next;
};
typedef struct stack *stk;
stk top=NULL;
stk push (int x)
 {
stk temp;
temp=(stk) malloc (sizeof (struct stack));
if (temp==NULL)
 {
printf("Stack is overflow.\n");
}
else
{
temp->data=x;
temp->next=top;
top=temp;
printf("Successfully pushed.\n");
}
}
stk pop()
{
stk temp;
if (top==NULL) {
printf("Stack is underflow.\n");
}
else
{
temp=top;
top=top->next;
printf("Popped value = %d\n", temp->data);
free(temp);
}
}
void display()
{
stk temp=top;
if (temp==NULL)
{
printf("Stack is empty.\n");
}
else
{
```

```
printf("Elements of the stack are : ");
while (temp!=NULL)
 {
printf("%d ", temp->data);
temp=temp->next;
}
printf("\n");
}
}
void isEmpty()
 {
stk temp=top;
if (top==NULL)
{
printf("Stack is empty.\n");
} else {
printf("Stack is not empty.\n");
}
}
void peek()
{
stk temp=top;
if (top==NULL)
{
printf("Stack is underflow.\n");
}
else
{
temp=top;
printf("Peek value = %d\n", temp->data);
}
}
```

INPUT:-

OUTPUT:-

RESULT:-

7. Write a C program to implement different Operations on Queue using Array representation

Aim: Write a program to implement queue using arrays.

Source Code:

```
#include <conio.h>
#include  <stdio.h>
int main()
{
int op, x;
while(1)
{
printf("1.Enqueue 2.Dequeue 3.Display 4.Is Empty 5.Size 6.Exit\n");
printf("Enter your option : ");
scanf("%d",&op);
switch(op)
{
case 1:
printf("Enter element : ");
scanf("%d",&x);
enqueue(x);
break;
case 2:
dequeue();
break;
case 3:
display();
break;
case 4:
isEmpty();
break;
case 5:
size();
break;
case 6: exit(0);
}
}
}
#define MAX 10
int queue[MAX],front=-1,rear=-1;
void dequeue()
{
int m;
```

```
if (front !=-1)
{
m=queue[front];
if (rear==front)
 {
front=-1;
rear=-1;
}
 else
{
front++;
}
printf("Deleted element = %d\n", m);
}
else
{
printf("Queue is underflow.\n");
}
}
void size()
{
if (front==-1 && rear==-1)
{
printf("Queue size : 0\n");
}
else
{
printf("Queue size : %d\n", rear-front+1);
}
}
void isEmpty()
{
if (front==-1 && rear==-1)
{
printf("Queue is empty.\n");
}
else
{
printf("Queue is not empty.\n");
}
}
void enqueue(int x)
 {
if (rear==MAX-1)
```

```
{
printf("Queue is overflow.\n");
}
 else
{
rear++;
queue[rear]=x;
printf("Successfully inserted.\n");
}
if (front==-1)
 {
front++;
}
}
void display()
 {
if (front==-1 && rear==-1)
{
printf("Queue is empty.\n");
}
else
 {
printf("Elements in the queue : ");
for (int i=front;i<=rear;i++)
{
printf("%d ", queue[i]);
}
printf("\n");
}
}
```

INPUT:-

OUTPUT:-

RESULT:-

8. Write a C program to implement different Operations on Queue using Linked Lists

AIM:- Write a program to implement queue using linked lists.

Source Code:

```
#include <conio.h>
#include  <stdio.h>
int main()
 {
int op, x;
while(1)
 {
printf("1.Enqueue 2.Dequeue 3.Display 4.Is Empty 5.Size 6.Exit\n");
printf("Enter your option : ");
scanf("%d",&op);
switch(op)
 {
case 1:
printf("Enter element : ");
scanf("%d",&x);
enqueue(x);
break;
case 2:
dequeue();
break;
case 3:
display();
break;
case 4:
isEmpty();
break;
case 5:
size();
break;
case 6: exit(0);
}
}
}
struct queue
{
int data;
struct queue *next;
};
```

```
typedef struct queue *Q;
Q front = NULL,rear=NULL;
void enqueue(int element)
{
Q temp=NULL;
temp=(Q)malloc (sizeof(struct queue));
if (temp==NULL)
{
printf("Queue is overflow.\n");
}
else
{
temp->data=element;
temp->next=NULL;
if (front==NULL)
{
front=temp;
}
else
{
rear->next=temp;
}
rear=temp;
printf("Successfully inserted.\n");
}
}
void dequeue()
{
if (front==NULL)
{
printf("Queue is underflow.\n");
}
else
{
Q temp=front;
if (front==rear)
{
front=rear->next;
}
else
{
front=front->next;
}
printf("Deleted value = %d\n", temp->data);
```

```
free(temp);
}
}
```

INPUT:-

OUTPUT:-

RESULT:-

9. Program to insert into BST and traversal using In-order, Pre-order and Post-order

Aim:

Write a program to create a binary search tree of integers and perform the following operations

1. insert a node
2. in-order traversal
3. pre-order traversal
4. post-order traversal

Source Code:

```
#include<stdio.h>
#include<stdlib.h>
void main()
{
int x, op;
BSTNODE root = NULL;
while(1)
 {
printf("1.Insert 2.Inorder Traversal 3.Preorder Traversal 4.Postorder Traversal5.Exit\n");
printf("Enter your option : ");
scanf("%d", &op);
switch(op)
{
case 1: printf("Enter an element to be inserted : ");
scanf("%d", &x);
root = insertNodeInBST(root,x);
break;
case 2:
if(root == NULL)
 {
printf("Binary Search Tree is empty.\n");
}
else
{
printf("Elements of the BST (in-order traversal): ");
inorderInBST(root);
printf("\n");
}
break;
case 3:
if(root == NULL)
{
printf("Binary Search Tree is empty.\n");
```

```
}
else
{
printf("Elements of the BST (pre-order traversal): ");
preorderInBST(root);
printf("\n");
}
break;
case 4:
if(root == NULL)
{
printf("Binary Search Tree is empty.\n");
}
else
{
printf("Elements of the BST (post-order traversal): ");
postorderInBST(root);
printf("\n");
}
break;
case 5:
exit(0);
}
}
}
struct node
 {
int data;
struct node *left, *right;
};
typedef struct node *BSTNODE;
BSTNODE newNodeInBST(int item)
{
BSTNODE temp = (BSTNODE)malloc(sizeof(struct node));
temp->data = item;
temp->left = temp->right = NULL;
return temp;
}
void inorderInBST(BSTNODE root)
 {
if(root!=NULL)
{
inorderInBST(root->left);
printf("%d ",root->data);
```

```
inorderInBST(root->right);
}
}
void preorderInBST(BSTNODE root)
 {
if(root!=NULL)
{
printf("%d ",root->data);
preorderInBST(root->left);
preorderInBST(root->right);
}
}
void postorderInBST(BSTNODE root)
{
if(root!=NULL)
{
postorderInBST(root->left);
postorderInBST(root->right);
printf("%d ",root->data);
}
}
BSTNODE insertNodeInBST(BSTNODE node, int ele)
{
if (node == NULL)
 {
printf("Successfully inserted.\n");
return newNodeInBST(ele);
}
if (ele < node->data)
node->left = insertNodeInBST(node->left,ele);
else if (ele > node->data)
node->right = insertNodeInBST(node->right,ele);
else
printf("Eleents already exist in BST.\n");
return node;
}
```

INPUT:-

OUTPUT:-

RESULT:-

10. Graph traversals implementation - Breadth First Search

AIM: Breadth First Search

BFS traversal of a graph, produces a spanning tree as final result. Spanning Tree is a graph without any loops. We use Queue data structure with maximum size of total number of vertices in the graph to implement BFS traversal of a graph.

Source Code:

```
#include <stdio.h>
#include <stdlib.h>
#define MAX 99
struct node
{
struct node *next;
int vertex;
};
typedef struct node * GNODE;
GNODE graph[20];
int visited[20];
int queue[MAX], front = -1,rear = -1;
int n;
void insertQueue(int vertex)
{
if (rear==MAX-1)
printf("Queue overflow\n");
else
 {
if (front==-1)
front=0;
rear=rear+1;
queue[rear]=vertex;
}
}
int isEmptyQueue()
{
if (front==-1 || front>rear)
return 1;
else
return 0;
}
int deleteQueue()
{
int deleteItem;
if (front==-1 || front>rear)
```

```
{
printf("Queue Underflow.\n");
exit(0);
}
deleteItem=queue[front];
front=front+1;
return deleteItem;
}
void BFS(int v)
{
int w;
insertQueue(v);
while(!isEmptyQueue())
{
v=deleteQueue(v);
printf("\n%d",v);
visited[v]=1;
GNODE g=graph[v];
for (;g!=NULL;g=g->next)
{
w=g->vertex;
if (visited[w]==0)
 {
insertQueue(w);
visited[w]=1;
}
}
}
printf("\n");
}
void main()
{
int N, E, s, d, i, j, v;
GNODE p, q;
printf("Enter the number of vertices : ");
scanf("%d",&N);
printf("Enter the number of edges : ");
scanf("%d",&E);
for(i=1;i<=E;i++)
{
printf("Enter source : ");
scanf("%d",&s);
printf("Enter destination : ");
scanf("%d",&d);
```

```
q=(GNODE)malloc(sizeof(struct node));
q->vertex=d;
q->next=NULL;
if(graph[s]==NULL)
 {
graph[s]=q;
} else {
p=graph[s];
while(p->next!=NULL)
p=p->next;
p->next=q;
}
}
for(i=1;i<=n;i++)
visited[i]=0;
printf("Enter Start Vertex for BFS : ");
scanf("%d", &v);
printf("BFS of graph : ");
BFS(v);
printf("\n");
}
```

INPUT:-

OUTPUT:-

RESULT:-

11. Graph traversals implementation - Depth First Search

Aim: Graph traversals

Graph traversal is technique used for searching a vertex in a graph. The graph traversal is decide the order of visiting of the vertices in the search process. A graph traversal finds the edges to be used in the search process without creating loops that means using graph traversal we visit all vertices of graph without getting into looping path.

Source Code:

```
#include<stdio.h>
#include<stdlib.h>
struct node
{
struct node *next;
int vertex;
};
typedef struct node * GNODE;
GNODE graph[20];
int visited[20];
int n;
void DFS(int i)
{
GNODE p;
printf("\n%d",i);
p=graph[i];
visited[i]=1;
while(p!=NULL)
{
i=p->vertex;
if (!visited[i])
DFS(i);
p=p->next;
}
}
void main()
{
int N,E,i,s,d;
GNODE q,p;
printf("Enter the number of vertices : ");
scanf("%d",&N);
printf("Enter the number of edges : ");
scanf("%d",&E);
for(i=1;i<=E;i++)
{
```

```
printf("Enter source : ");
scanf("%d",&s);
printf("Enter destination : ");
scanf("%d",&d);
q=(GNODE)malloc(sizeof(struct node));
q->vertex=d;
q->next=NULL;
if(graph[s]==NULL)
graph[s]=q;
else {
p=graph[s];
while(p->next!=NULL)
p=p->next;
p->next=q;
}
}
for(i=0;i<n;i++)
visited[i]=0;
printf("Enter Start Vertex for DFS : ");
int v;
scanf("%d",&v);
printf("DFS of graph : ");
DFS(v);
printf("\n");
}
```

INPUT:-

OUTPUT:-

RESULT:-

12. Problem on travelling sales man

Aim: The problem statement is given as follows. Solve the problem using branch and bound technique
A traveler needs to visit all the cities from a given list of cities, where distances between all the cities are known
and each city should be visited exactly once. What is the shortest possible route that he visits each city exactly
once and returns to the origin city.

Source Code:

```
#include<stdio.h>
int matrix[25][25], visitedCities[10], limit_cities, cost = 0;
int travellingcities(int c)
{
int count, nearestCity = 999;
int minimum = 999, temp = 0;
for(count = 0; count < limit_cities; count++)
{
if((matrix[c][count] != 0) && (visitedCities[count] == 0))
{
if(matrix[c][count] < minimum)
{
minimum = matrix[count][0] + matrix[c][count];
}
temp = matrix[c][count];
nearestCity = count;
}
}
if(minimum != 999)
{
cost = cost + temp;
}
return nearestCity;
}
void minimum_cost(int city)
{
// find the minimum cost to reach the nearestcity and return it
int nearestCity;
visitedCities[city] = 1;
printf("%d ", city + 1);
nearestCity = travellingcities(city);
if(nearestCity == 999)
{
nearestCity = 0;
printf("%d", nearestCity + 1);
```

```
cost = cost + matrix[city][nearestCity];
return;
}
minimum_cost(nearestCity);
}
int main()
{
int i, j;
printf("Enter Total Number of Cities: ");
scanf("%d",  &limit_cities);
printf("Enter Cost Matrix\n");
for(i = 0; i < limit_cities; i++)
{
printf("Enter %d Elements in row[%d]\n", limit_cities, i + 1);
for(j = 0; j < limit_cities; j++)
{
scanf("%d", &matrix[i][j]);
}
visitedCities[i] = 0;
}
printf("Entered Cost Matrix");
for(i = 0; i < limit_cities; i++)
{
printf("\n");
for(j = 0; j < limit_cities; j++)
{
printf("%d ", matrix[i][j]);
}
}
printf("\n");
printf("Path: ");
minimum_cost(0);
printf("\n");
printf("Minimum Cost is %d ", cost);
return 0;
}
```

INPUT:-

OUTPUT:-

RESULT:-

13. File operations

13.A)

AIM:-Follow the instructions given below to write a program to open a file and to print its contents on the screen.

Source Code:

```
#include <stdio.h>
void main()
{
FILE *fp;
char ch;
fp = fopen("SampleText1.txt", "w");
printf("Enter the text with @ at end : ");
while ((ch = getchar()) != '@')
 {
putc(ch, fp);
}
putc(ch, fp);
fclose(fp);
fp = fopen("SampleText1.txt", "r");
printf("Given message is : ");
while ((ch = getc(fp)) != '@')
{
putchar(ch);
}
printf("\n");
fclose(fp);
}
```

INPUT:-

OUTPUT:-

RESULT:-

13.B) Write a C program to Copy contents of one File into another File

Aim: Write a program to copy contents of one file into another file. Follow the instructions given below to write a program to copy the contents of one file to another file: Open a new file " SampleTextFile1.txt " in write mode Write the content onto the file

- Close the file
- Open an existing file " SampleTextFile1.txt " in read mode
- Open a new file " SampleTextFile2.txt " in write mode
- Copy the content from existing file to new file
- Close the files
- Open the copied file in read mode
- Read the text from file and print on the screen
- Close the file

Source Code:

```
#include <stdio.h>
void main() {
FILE *fp, *fp1, *fp2;
char ch;
fp = fopen("SampleTextFile1.txt", "w");
printf("Enter the text with @ at end : ");
while ((ch = getchar()) != '@') {
putc(ch, fp);
}
putc(ch, fp);
fclose(fp);
fp1 = fopen("SampleTextFile1.txt", "r");
fp2 = fopen("SampleTextFile2.txt", "w");
while ((ch = getc(fp1)) != '@') {
putc(ch, fp2);
}
putc(ch, fp2);
fclose(fp1);
fclose(fp2);
fp2 = fopen("SampleTextFile2.txt", "r");
printf("Copied text is : ");
while ((ch = getc(fp2)) != '@') {
putchar(ch);
}
printf("\n");
fclose(fp2);
}
```

INPUT:-

OUTPUT:-

RESULT:-

13.C) Write a C program to Merge two Files and stores their contents in another File

Aim: Write a program to merge two files and stores their contents in another file.

Open a new file " SampleDataFile1.txt " in write mode Write the content onto the file Close the file
Open another new file " SampleDataFile2.txt " in write mode Write the content onto the file

- Close the file
- Open first existing file " SampleDataFile1.txt " in read mode
- Open a new file " SampleDataFile3.txt " in write mode
- Copy the content from first existing file to new file
- Close the first existing file
- Open another existing file " SampleDataFile2.txt " in read mode
- Copy its content from existing file to new file
- Close that existing file
- Close the merged file

Source Code:

```
#include <stdio.h>
void main() {
FILE *fp1, *fp2, *fp3;
char ch;
fp1 = fopen("SampleDataFile1.txt", "w");
printf("Enter the text with @ at end for file-1 :\n");
while ((ch = getchar()) != '@') {
putc(ch, fp1);
```

```
}
putc(ch, fp1);
fclose(fp1);
fp2 = fopen("SampleDataFile2.txt", "w");
printf("Enter the text with @ at end for file-2 :\n");
while ((ch = getchar()) != '@') {
putc(ch, fp2);
}
putc(ch, fp2);
fclose(fp2);
fp1 = fopen("SampleDataFile1.txt", "r");
fp3 = fopen("SampleDataFile3.txt", "w");
while ((ch = getc(fp1)) != '@') {
putc(ch, fp3);
}
fclose(fp1);
fp2 = fopen("SampleDataFile2.txt", "r");
while ((ch = getc(fp2)) != '@') {
putc(ch, fp3);
}
putc(ch, fp3);
fclose(fp2);
fclose(fp3);
fp3 = fopen("SampleDataFile3.txt", "r");
printf("Merged text is : ");
while ((ch = getc(fp3)) != '@') {
putchar(ch);
}
printf("\n");
fclose(fp3);
}
```

Input:-

output:-

Result:-

13.D) Write a C program to Delete a File

Aim: Write a program to delete a file.

Source Code:

```
#include <stdio.h>
void main()
{
FILE *fp;
int status;
char fileName[40], ch;
printf("Enter a new file name : ");
gets(fileName);
fp = fopen(fileName, "w");
printf("Enter the text with @ at end : ");
while ((ch = getchar()) != '@')
 {
putc(ch, fp);
}
putc(ch, fp);
fclose(fp);
fp = fopen(fileName, "r");
printf("Given message is : ");
while ((ch = getc(fp)) != '@')
{
putchar(ch);
}
printf("\n");
fclose(fp);
status = remove(fileName);
if (status == 0)
printf("%s file is deleted successfully\n", fileName);
else
 {
printf("Unable to delete the file -- ");
perror("Error\n");
}
}
```

Input:-

Output:-

Result:-

13.E) Write a C program to Copy last n characters from one File to another File

Aim: Write a program to copy last n characters from file-1 to file-2.

- open a new file " TestDataFile1.txt " in write mode
- write the content onto the file
- close the file
- open an existing file " TestDataFile1.txt " in read mode
- open a new file " TestDataFile2.txt " in write mode
- read the number of characters to copy
- set the cursor position by using fseek()
- copy the content from existing file to new file
- close the files
- open the copied file " TestDataFile2.txt " in read mode
- read the text from file and print on the screen
- close the file

Source Code:

```
#include <stdio.h>
void main()
{
FILE *fp, *fp1, *fp2;
int num, length;
char ch;
fp = fopen("TestDataFile1.txt", "w");
printf("Enter the text with @ at end : ");
while ((ch = getchar()) != '@')
{
putc(ch, fp);
}
putc(ch, fp);
fclose(fp);
fp1 = fopen("TestDataFile1.txt", "r");
fp2 = fopen("TestDataFile2.txt", "w");
printf("Enter number of characters to copy : ");
scanf("%d", &num);
fseek(fp1, 0L, SEEK_END);
length = ftell(fp1);
fseek(fp1, (length - num - 1), SEEK_SET);
while ((ch = getc(fp1)) != '@')
{
putc(ch, fp2);
```

```
}
putc(ch, fp2);
fclose(fp1);
fclose(fp2);
fp2 = fopen("TestDataFile2.txt", "r");
printf("Copied text is : ");
while ((ch = getc(fp2)) != '@')
{
putchar(ch);
}
printf("\n");
fclose(fp2);
}
```

Input:-

output:-

Result:-

14.