LAB 1
08/01/2024

QUESTION:

Write and initialize an integer array of size n perform following operations on array
Print Minimum,maximum,second
Maximum,average,mode,sort,reverse,search,frequency.

SOLUTION:

INPUT:

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

//FUNCTION FOR MODE
int arr_mode(int arr[],int n)
{
    int maxcount = 0, mode = -1;
    for(int i=0;i<n;i++)
    {
        int count = 0;
        for(int j=i+1;j<n;j++)
        {
            if(arr[j] == arr[i])
            {
                count++;
            }
        }
        if(count > maxcount){
            maxcount = count;
            mode = arr[i];
        }
    }
    return mode;
}
```

```c
//FUNCTION FOR SWAP
void swap(int *x,int *y)
{
    int temp = *x;
    *x = *y;
    *y = temp;
}

//FUNCTION FOR SORT
void sort(int arr[],int n)
{
    for(int i=0;i<n-1;i++)
    {
        for(int j=0;j<n-1-i;j++)
        {
            if(arr[j] > arr[j+1])
            {
                swap(&arr[j],&arr[j+1]);
            }
        }
    }
}

//FUNCTION FOR PRINT SORTED ARRAY
void printArray(int arr[], int size) {
    for (int i = 0; i < size; i++) {
        printf("%d ", arr[i]);
    }
    printf("\n");
}

//FUNCTION FOR REVERSE ARRAY
int rev(int arr[],int n)
{
```

```
    int k = n-1;
    for(int i=0;i<n/2;i++)
    {
        int temp = arr[i];
        arr[i] = arr[k];
        arr[k] = temp;
        k--;
    }
}

// FUNCTION TO SEARCH FOR THE INDEX OF AN ELEMENT IN AN ARRAY
int searchElementIndex(int array[], int size, int target) {
    for (int i = 0; i < size; i++)
    {
        if (array[i] == target)
        {
            return i;
        }
    }
    return -1;
}

int main()
{
    time_t t = time(NULL);
    struct tm tm = *localtime(&t);
    printf("Ram Krishna BT23CSE026\n");
    printf("%d-%02d-%02d %02d:%02d:%02d\n", tm.tm_year + 1900, tm.tm_mon + 1,
tm.tm_mday, tm.tm_hour, tm.tm_min, tm.tm_sec);
    printf("********************************\n");
    int n,min,max,i,target,location = 1,sum=0;
    float average;
    printf("Enter the value of n : ");
    scanf("%d",&n);
    int arr[n];
```

```
    printf("Enter the elements of the array : \n");
    for(int i=0;i<n;i++)
    {
        printf("Enter element at index %d: ", i);
        scanf("%d",&arr[i]);
    }

    //MINIMUM ELEMENT IN THE ARRAY
    min = arr[0];
    for(i=1;i<n;i++)
    {
        if(arr[i]<min)
        {
            min = arr[i];
            location = i+1;
        }
    }
    printf("Minimum element is present at the location %d and it's value is %d
\n",location,min);

    //MAXIMUM ELEMENT IN THE ARRAY
    max = arr[0];
    for(i=1;i<n;i++)
    {
        if(arr[i]>max)
        {
            max = arr[i];
            location = i+1;
        }
    }
    printf("Maximum element is present at the location %d and it's value is %d
\n",location,max);

    //AVERAGE OF ELEMENTS IN THE ARRAY
    for(i=0;i<n;i++)
```

```
{
    sum += arr[i];
    average = (sum/n)+(sum%n);
}
printf("Average of the array is : %f \n",average);

//MODE IN THE ARRAY
int mode = arr_mode(arr,n);
if(mode != -1)
{
    printf("The mode of the array is : %d \n",mode);
}
else
{
    printf("No mode found \n");
}

//SORTING AND PRINTING SORTED ARRAY
sort(arr,n);
printf("Sorted array: \n");
printArray(arr, n);

//PRINTING REVERSE ARRAY
rev(arr,n);
printf("Reverse array: \n");
printArray(arr, n);

//PRINTING SEARCHING ELEMENT INDEX
printf("enter the searching element u want : \n");
scanf("%d",&target);
int index = searchElementIndex(arr, n, target);
if (index != -1) {
    printf("Element %d found at index %d.\n", target, index);
} else {
    printf("Element %d not found in the array.\n", target);
```

```
  }

  return 0;
}
```

OUTPUT:

```
PROBLEMS    OUTPUT    TERMINAL    DEBUG CONSOLE    PORTS                                    Code - DSA

PS C:\Users\MANSI\Dropbox\BT23CSE026> cd "c:\Users\MANSI\Dropbox\BT23CSE026\DSA LAB\" ; if ($?) { gcc LAB1.c -o LAB1 } ; if ($?) { .\LAB1 }
Ram Krishna BT23CSE026
2024-02-18 22:55:35
********************************
Enter the value of n : 5
Enter the elements of the array :
Enter element at index 0: 8
Enter element at index 1: 3
Enter element at index 2: 11
Enter element at index 3: 6
Enter element at index 4: 7
Minimum element is present at the location 2 and it's value is 3
Maximum element is present at the location 3 and it's value is 11
Average of the array is : 7.000000
No mode found
Sorted array:
3 6 7 8 11
Reverse array:
11 8 7 6 3
enter the searching element u want :
7
Element 7 found at index 2.
PS C:\Users\MANSI\Dropbox\BT23CSE026\DSA LAB>
```

LAB 2
15/01/2024

QUESTION:

1.Create an array and input a new element at start, at end and at a specific index.
2.Create an array and delete an element from start, from end and from a specific index.
3.Input an array and search for an element using binary search and linear search.
4.Input an array and sort it using bubble sort.

SOLUTION:

1.

INPUT:

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

// FUNCTION FOR INSERT ELEMENT AT INDEX
int insertAt(int arr[], int size, int insert, int index)
```

```
{
    for (int i = size - 1; i >= index; i--)
    {
        arr[i + 1] = arr[i];
    }
    arr[index] = insert;
    return size + 1;
}

int main()
{
    int n;
    time_t t = time(NULL);
    struct tm tm = *localtime(&t);
    printf("Ram Krishna BT23CSE026\n");
    printf("%d-%02d-%02d %02d:%02d:%02d\n", tm.tm_year + 1900, tm.tm_mon + 1,
tm.tm_mday, tm.tm_hour, tm.tm_min, tm.tm_sec);
    printf("********************************\n");
    printf("Enter size of array : ");
    scanf("%d", &n);
    int arr[100];
    printf("Enter array elements at :\n");
    for (int i = 0; i < n; i++)
    {
        printf("arr[%d] : ", i);
        scanf("%d", &arr[i]);
    }
    printf("Enter the new element at index : ");
    int i;
    scanf("%d", &i);
    printf("Enter the element to be inserted : ");
    int num;
    scanf("%d", &num);
    n = insertAt(arr, n, num, i);
    printf("\nModified Array is : \n");
```

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

OUTPUT:

```
PROBLEMS    OUTPUT    TERMINAL    DEBUG CONSOLE    PORTS

PS C:\Users\MANSI\Dropbox\BT23CSE026> cd "c:\Users\MANSI\Dropbox\BT23CSE026\DSA LAB\" ; if ($?) { gcc tempCodeRunnerFile.c -o te
erFile }
Ram Krishna BT23CSE026
2024-02-18 22:58:50
*********************************
Enter size of array : 6
Enter array elements at :
arr[0] : 7
arr[1] : 2
arr[2] : 6
arr[3] : 4
arr[4] : 12
arr[5] : 10
Enter the new element at index : 4
Enter the element to be inserted : 15

Modified Array is :
7 2 6 4 15 12 10
PS C:\Users\MANSI\Dropbox\BT23CSE026\DSA LAB>
```

2.

INPUT:

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

// FUNCTION FOR DELETE ELEMENT AT ANY INDEX
int deleteAt(int arr[], int size, int deleteIndex)
{
    for (int i = deleteIndex; i<size;i++)
    {
        arr[i] = arr[i+1];
    }
    return size - 1;
}

int main()
{
```

```
    int n;
    time_t t = time(NULL);
    struct tm tm = *localtime(&t);
    printf("Ram Krishna BT23CSE026\n");
    printf("%d-%02d-%02d %02d:%02d:%02d\n", tm.tm_year + 1900, tm.tm_mon + 1,
tm.tm_mday, tm.tm_hour, tm.tm_min, tm.tm_sec);
    printf("*********************************\n");
    printf("Enter size of array : ");
    scanf("%d", &n);
    int arr[100];
    printf("Enter array elements at :\n");
    for (int i = 0; i < n; i++)
    {
        printf("arr[%d] : ", i);
        scanf("%d", &arr[i]);
    }
    printf("Delete element from index : ");
    int i;
    scanf("%d", &i);
    n = deleteAt(arr, n, i);
    printf("\nModified Array is : \n");
    for (int i = 0; i < n; i++)
    {
        printf("%d ", arr[i]);
    }
    printf("\n");
    return 0;
}
```

OUTPUT:

```
PROBLEMS    OUTPUT    TERMINAL    DEBUG CONSOLE    PORTS

PS C:\Users\MANSI\Dropbox\BT23CSE026> cd "c:\Users\MANSI\Dropbox\BT23CSE026\DSA LAB\" ; if ($?) { gcc tempC
erFile }
Ram Krishna BT23CSE026
2024-02-18 23:00:13
********************************
Enter size of array : 5
Enter array elements at :
arr[0] : 8
arr[1] : 3
arr[2] : 5
arr[3] : 6
arr[4] : 12
Delete element from index : 2

Modified Array is :
8 3 6 12
PS C:\Users\MANSI\Dropbox\BT23CSE026\DSA LAB>
```

3.

INPUT:

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

// FUNCTION FOR BINARY SEARCH
int binarySearch(int arr[], int size, int search)
{
    int i=-1;
    int low=0;
    int high=size-1;
    while(low<=high)
    {
        int mid=(low+high)/2;
        if(arr[mid]==search)
        {
            i=mid;
            break;
        }
        else if(arr[mid]>search)
        {
            high=mid-1;
        }
        else{
```

```
            low=mid+1;
        }
    }
    return i;
}

// FUNCTION FOR LINEAR SEARCH
int linearSearch(int arr[],int size,int search){
    for (int i = 0; i < size; ++i)
    {
        if(arr[i]==search)
        {
            return i;
        }
    }
    return -1;
}

int main()
{
    int n;
    time_t t = time(NULL);
    struct tm tm = *localtime(&t);
    printf("Ram Krishna BT23CSE026\n");
    printf("%d-%02d-%02d %02d:%02d:%02d\n", tm.tm_year + 1900, tm.tm_mon + 1,
tm.tm_mday, tm.tm_hour, tm.tm_min, tm.tm_sec);
    printf("*********************************\n");
    printf("Enter size of array : ");
    scanf("%d", &n);
    int arr[100];
    printf("Enter array elements at :\n");
    for (int x = 0; x < n; x++)
    {
        printf("arr[%d] : ", x);
        scanf("%d", &arr[x]);
```

```
    }
    printf("Enter element to search : ");
    int search;
    scanf("%d", &search);
    int i=binarySearch(arr,n,search);
    printf("Element found at index from Binary Search: %d\n",i);
    i=linearSearch(arr,n,search);
    printf("Element found at index from Linear Search: %d\n",i);
    printf("\n");
    return 0;
}
```

OUTPUT:

```
PROBLEMS    OUTPUT    TERMINAL    DEBUG CONSOLE    PORTS

PS C:\Users\MANSI\Dropbox\BT23CSE026> cd "c:\Users\MANSI\Dropbox\BT23CSE026\DSA LAB\" ; if ($?) { gcc LAB2.c -o LAB2 } ; if (
Ram Krishna BT23CSE026
2024-02-18 23:01:30
********************************
Enter size of array : 6
Enter array elements at :
arr[0] : 2
arr[1] : 8
arr[2] : 9
arr[3] : 15
arr[4] : 19
arr[5] : 21
Enter element to search : 15
Element found at index from Binary Search: 3
Element found at index from Linear Search: 3

PS C:\Users\MANSI\Dropbox\BT23CSE026\DSA LAB>
```

4.

INPUT:

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

// FUNCTION FOR BUBBLE SORT
void bubbleSort(int arr[], int size)
{
    for(int i=0;i<size-1;i++){
        for(int j=0;j<size-1-i;j++){
            if(arr[j+1]<arr[j]){
                int t=arr[j];
                arr[j]=arr[j+1];
```

```
                arr[j+1]=t;
            }
        }
    }
}

int main()
{
    int n;
    time_t t = time(NULL);
    struct tm tm = *localtime(&t);
    printf("Ram Krishna BT23CSE026\n");
    printf("%d-%02d-%02d %02d:%02d:%02d\n", tm.tm_year + 1900, tm.tm_mon + 1,
tm.tm_mday, tm.tm_hour, tm.tm_min, tm.tm_sec);
    printf("*********************************\n");
    printf("Enter size of array : ");
    scanf("%d", &n);
    int arr[100];
    printf("Enter array elements at :\n");
    for (int i = 0; i < n; i++)
    {
        printf("arr[%d] : ", i);
        scanf("%d", &arr[i]);
    }
    printf("Unsorted array : \n");
    for (int i = 0; i < n; i++){
        printf("%d ", arr[i]);
    }
    bubbleSort(arr,n);
    printf("\nSorted array : \n");
    for(int i=0;i<n;i++){
        printf("%d ",arr[i]);
    }
    printf("\n");
    return 0;
```

```
}
```

OUTPUT:

```
PROBLEMS    OUTPUT    TERMINAL    DEBUG CONSOLE    PORTS

PS C:\Users\MANSI\Dropbox\BT23CSE026> cd "c:\Users\MANSI\Dropbox\BT23CSE026\DSA LAB\" ; if ($?) { gc
Ram Krishna BT23CSE026
2024-02-18 23:02:45
********************************
Enter size of array : 6
Enter array elements at :
arr[0] : 2
arr[1] : 6
arr[2] : 9
arr[3] : 1
arr[4] : 12
arr[5] : 17
Unsorted array :
2 6 9 1 12 17
Sorted array :
1 2 6 9 12 17
PS C:\Users\MANSI\Dropbox\BT23CSE026\DSA LAB>
```

LAB 3
29/01/2024

QUESTION:

1.Sort int array using insertion sort
2.Sort a given string using insertion sort.
3.Selection sort of 1 and 2.
5.Merge sort of 1 and 2.
7.Create a sorted array using n sorted arrays.
8.Sort an integer array using reverse insertion sort.

SOLUTION:

1.

INPUT:

```
//SORT INTEGER ARRAY USING INSERTION SORT.

#include<stdio.h>
#include<time.h>

//FUNCTION FOR INSERTION SORT
 void insertionSort(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;
    }
}

int main()
{
    time_t t = time(NULL);
    struct tm tm = *localtime(&t);
    printf("Ram Krishna BT23CSE026\n");
    printf("%d-%02d-%02d %02d:%02d:%02d\n", tm.tm_year + 1900, tm.tm_mon + 1,
tm.tm_mday, tm.tm_hour, tm.tm_min, tm.tm_sec);
    printf("********************************\n");
    int key = 9;
    int arr[] = {6,7,25,14,8,16,15};
    int n = sizeof(arr)/sizeof(arr[0]);
    printf("Array after using insertion array : ");
    insertionSort(arr,n);
    for(int i=0;i<n;i++)
    {
        printf("%d ",arr[i]);
    }
    return 0;
}
```

OUTPUT:

```
PS C:\Users\MANSI\Dropbox\BT23CSE026> cd "c:\Users\MANSI\Dropbox\BT23CSE026\DSA LAB\" ; if ($?) { gcc LAB3.c -o LAB3 } ; if ($?) {
Ram Krishna BT23CSE026
2024-02-18 23:06:56
********************************
Array after using insertion array : 6 7 8 14 15 16 25
PS C:\Users\MANSI\Dropbox\BT23CSE026\DSA LAB>
```

2.

INPUT:

```
//SORT INTEGER ARRAY USING SELECTION SORT.
#include <stdio.h>
#include <time.h>

//FUNCTION FOR SWAPPING
void swap(int *x, int *y)
{
   int temp = *x;
   *x = *y;
   *y = temp;
}

 //FUNCTION FOR SELECTION SORT
void selectionSort(int arr[], int n)
{
   int i, j, min;
   for (i = 0; i < n-1; i++)
   {
      min = i;
      for (j = i+1; j < n; j++)
        if (arr[j] < arr[min])
        {
          min = j;
        }
         if(min != i)
         {
          swap(&arr[min], &arr[i]);
         }
```

```
    }
}

//FUNCTION FOR PRINT ARRAY
void printArray(int arr[], int size)
{
    int i;
    for (i=0; i < size; i++)
        printf("%d ", arr[i]);
    printf("\n");
}
int main()
{
    time_t t = time(NULL);
    struct tm tm = *localtime(&t);
    printf("Ram Krishna BT23CSE026\n");
    printf("%d-%02d-%02d %02d:%02d:%02d\n", tm.tm_year + 1900, tm.tm_mon + 1,
tm.tm_mday, tm.tm_hour, tm.tm_min, tm.tm_sec);
    printf("*********************************\n");
    int arr[] = {64, 25, 12, 22, 11 ,7 ,13};
    int n = sizeof(arr)/sizeof(arr[0]);
    selectionSort(arr, n);
    printf("Sorted array: \n");
    printArray(arr, n);
    return 0;
}
```

OUTPUT:

```
PROBLEMS    OUTPUT    TERMINAL    DEBUG CONSOLE    PORTS                                   Code - DSA LAB

PS C:\Users\MANSI\Dropbox\BT23CSE026> cd "c:\Users\MANSI\Dropbox\BT23CSE026\DSA LAB\" ; if ($?) { gcc tempCodeRunnerFile.c -o tempCodeRunnerFile } ; if ($?) { .\tempCo
erFile }
Ram Krishna BT23CSE026
2024-02-18 23:08:06
*********************************
Sorted array:
7 11 12 13 22 25 64
PS C:\Users\MANSI\Dropbox\BT23CSE026\DSA LAB>
```

3.

INPUT:

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

```c
#include <time.h>

void insertionSort(char arr[],int n)
{
    for(int i=1;i<n;i++)
    {
        char t=arr[i];
        int k=i-1;
        for(int j=i-1;j>=0;j--)
        {
            if(arr[k]<t)
            {
            break;
            }
            arr[j+1]=arr[j];
            k--;
        }
        arr[k+1]=t;
    }
}

int main()
{
    time_t t = time(NULL);
    struct tm tm = *localtime(&t);
    printf("Ram Krishna BT23CSE026\n");
    printf("%d-%02d-%02d %02d:%02d:%02d\n", tm.tm_year + 1900, tm.tm_mon + 1,
tm.tm_mday, tm.tm_hour, tm.tm_min, tm.tm_sec);
    printf("**********************************\n");
    char arr[]="datastructure";
    int n = 13;
    insertionSort(arr,n);
    printf("%s\n",arr);
    return 0;
}
```

OUTPUT:

```
PROBLEMS    OUTPUT    TERMINAL    DEBUG CONSOLE    PORTS

PS C:\Users\MANSI\Dropbox\BT23CSE026> cd "c:\Users\MANSI\Dropbox\BT23CSE026\DSA LAB\" ; if ($?) {
Ram Krishna BT23CSE026
2024-02-18 23:14:09
*******************************
aacderrstttuu
PS C:\Users\MANSI\Dropbox\BT23CSE026\DSA LAB>
```

4.

INPUT:

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

void selectionSort(char arr[],int n)
{
    for(int i=0;i<n;i++)
    {
        int k=i;
        for(int j=i+1;j<n;j++)
        {
            if(arr[j]<arr[k])
            {
                k=j;
            }
        }
        char t=arr[i];
        arr[i]=arr[k];
        arr[k]=t;
    }
}

int main()
{
    time_t t = time(NULL);
    struct tm tm = *localtime(&t);
```

```
    printf("Ram Krishna BT23CSE026\n");
    printf("%d-%02d-%02d %02d:%02d:%02d\n", tm.tm_year + 1900, tm.tm_mon + 1,
tm.tm_mday, tm.tm_hour, tm.tm_min, tm.tm_sec);
    printf("*********************************\n");
    char arr[]="datastructure";
    int n = 13;
    selectionSort(arr,n);
    printf("%s\n",arr);
    return 0;
}
```

OUTPUT:

```
PROBLEMS    OUTPUT    TERMINAL    DEBUG CONSOLE    PORTS

PS C:\Users\MANSI\Dropbox\BT23CSE026> cd "c:\Users\MANSI\Dropbox\BT23CSE026\DSA LAB\" ; if ($?) { gcc LAB3.c -o
Ram Krishna BT23CSE026
2024-02-18 23:12:14
*********************************
aacderrstttuu
PS C:\Users\MANSI\Dropbox\BT23CSE026\DSA LAB>
```

5.

INPUT:

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

void merge(int a[],int l,int mid,int h)
{
    int b[h-l+1],i=l,j=mid+1,k=0;
    while(i<=mid&&j<=h)
    {
        if(a[i]<a[j])
        {
            b[k++]=a[i++];
        }
        else
        {
            b[k++]=a[j++];
        }
    }
```

```
    while(i<=mid)
    {
        b[k++]=a[i++];
    }
    while(j<=h)
    {
        b[k++]=a[j++];
    }
    for(int x=l,k=0;x<=h;x++,k++)
    {
        a[x]=b[k];
    }
}
void mergeSort(int a[],int l,int h)
{
    if(l<h)
    {
        int mid=(l+h)/2;
        mergeSort(a,l,mid);
        mergeSort(a,mid+1,h);
        merge(a,l,mid,h);
    }
}

int main()
{
    time_t t = time(NULL);
    struct tm tm = *localtime(&t);
    printf("Ram Krishna BT23CSE026\n");
    printf("%d-%02d-%02d %02d:%02d:%02d\n", tm.tm_year + 1900, tm.tm_mon + 1,
tm.tm_mday, tm.tm_hour, tm.tm_min, tm.tm_sec);
    printf("*********************************\n");
    int a[]={5,16,5,7,12};
    int n = 5;
    mergeSort(a,0,n-1);
```

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

OUTPUT:

```
PROBLEMS    OUTPUT    TERMINAL    DEBUG CONSOLE    PORTS

PS C:\Users\MANSI\Dropbox\BT23CSE026> cd "c:\Users\MANSI\Dropbox\BT23CSE026\DSA LAB\" ; if ($?) { gcc LAB3
Ram Krishna BT23CSE026
2024-02-18 23:15:53
********************************
5 5 7 12 16
PS C:\Users\MANSI\Dropbox\BT23CSE026\DSA LAB>
```

6.

INPUT:

```
//SORT STRING USING MERGE SORT.
#include <stdio.h>
#include <time.h>

void merge(char a[],int l,int mid,int h)
{
    char b[h-l+1],i=l,j=mid+1,k=0;
    while(i<=mid&&j<=h)
    {
        if(a[i]<a[j])
        {
            b[k++]=a[i++];
        }
        else
        {
            b[k++]=a[j++];
        }
    }
    while(i<=mid)
```

```
    {
        b[k++]=a[i++];
    }
    while(j<=h)
    {
        b[k++]=a[j++];
    }
    for(int x=l,k=0;x<=h;x++,k++)
    {
        a[x]=b[k];
    }
}

void mergeSort(char a[],int l,int h)
{
    if(l<h)
    {
        int mid=(l+h)/2;
        mergeSort(a,l,mid);
        mergeSort(a,mid+1,h);
        merge(a,l,mid,h);
    }
}

int main()
{
    time_t t = time(NULL);
    struct tm tm = *localtime(&t);
    printf("Ram Krishna BT23CSE026\n");
    printf("%d-%02d-%02d %02d:%02d:%02d\n", tm.tm_year + 1900, tm.tm_mon + 1,
tm.tm_mday, tm.tm_hour, tm.tm_min, tm.tm_sec);
    printf("*********************************\n");
    char a[]="datastructure";
    int n=13;
    mergeSort(a,0,n-1);
```

```
    printf("%s\n",a);
    return 0;
}
```

OUTPUT:

```
PROBLEMS    OUTPUT    TERMINAL    DEBUG CONSOLE    PORTS

Ram Krishna BT23CSE026
2024-02-18 23:17:54
**********************************
aacderrstttuu
PS C:\Users\MANSI\Dropbox\BT23CSE026\DSA LAB>
```

7.

INPUT:

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

void merge(int a[],int b[],int *c,int l1,int l2,int l3)
{
    int i=0,j=0,k=0;
    while(i<l1&&j<l2)
    {
        if(a[i]<b[j])
        {
            c[k++]=a[i++];
        }
        else
        {
            c[k++]=b[j++];
        }
    }
    while(i<l1)
    {
        c[k++]=a[i++];
    }
    while(j<l2)
    {
```

```
        c[k++]=b[j++];
    }
}

int main()
{
    time_t t = time(NULL);
    struct tm tm = *localtime(&t);
    printf("Ram Krishna\n");
    printf("%d-%02d-%02d %02d:%02d:%02d\n", tm.tm_year + 1900, tm.tm_mon + 1,
tm.tm_mday, tm.tm_hour, tm.tm_min, tm.tm_sec);
    printf("********************************\n");
    int n=5,m=5;
    int a[5][5]={ {5,8,13,17,28}, {2,2,13,19,30}, {0,7,9,13,15}, {2,13,25,35,40},
{0,22,33,54,79}};
    int size=m+m;
    int *ptr=(int*)malloc(size*sizeof(int));
    merge(a[0],a[1],ptr,m,m,size);
    for(int i=2;i<n;i++)
    {
        size+=m;
        int *temp = (int*)malloc((size)*sizeof(int));
        merge(ptr,a[i],temp,size-m,m,size);
        ptr=temp;
    }
    for(int i=0;i<n*m;i++)
    {
        printf("%d ",ptr[i]);
    }
    printf("\n");
    return 0;
}
```

OUTPUT:

```
PROBLEMS    OUTPUT    TERMINAL    DEBUG CONSOLE    PORTS

Ram Krishna
2024-02-18 23:19:12
*********************************
0 0 2 2 2 5 7 8 9 13 13 13 13 15 17 19 22 25 28 30 33 35 40 54 79
PS C:\Users\MANSI\Dropbox\BT23CSE026\DSA LAB>
```

8.

INPUT:

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

void insertionSort(int a[],int n)
{
    for(int i=n-2;i>=0;i--)
    {
        int t=a[i];
        int k=i+1;
        for(int j=i+1;j<n;j++)
        {
            if(t<a[k])
            {
                break;
            }
            a[j-1]=a[j];
            i++;
        }
        a[k-1]=t;
    }
}

int main()
{
    time_t t = time(NULL);
    struct tm tm = *localtime(&t);
    printf("Ram Krishna BT23CSE026\n");
```

```
    printf("%d-%02d-%02d %02d:%02d:%02d\n", tm.tm_year + 1900, tm.tm_mon + 1,
tm.tm_mday, tm.tm_hour, tm.tm_min, tm.tm_sec);
    printf("********************************\n");
    int a[]={11,15,18,21,26};
    int n=5;
    insertionSort(a,n);
    for(int i=0;i<n;i++)
    {
        printf("%d ",a[i]);
    }
    printf("\n");
    return 0;
}
```

OUTPUT:

```
PROBLEMS    OUTPUT    TERMINAL    DEBUG CONSOLE    PORTS

PS C:\Users\MANSI\Dropbox\BT23CSE026> cd "c:\Users\MANSI\Dropbox\BT23CSE026\DSA LAB\" ; if ($?) { gcc LAB3.c -o LAB3 } ;
Ram Krishna BT23CSE026
2024-02-18 23:20:20
********************************
11 15 18 21 26
PS C:\Users\MANSI\Dropbox\BT23CSE026\DSA LAB>
```

LAB 4
12/02/2024

QUESTION:

1.Create a linked list by inserting a node at the end of linked list and print it.
2.Create a linked list by inserting a node at the beginning of the linked list and print it.
3.Print the given singly linked list in reverse order.
4.count no. of nodes in linked list.
5.Reverse the singly linked list without recursion.
6.Reverse the singly linked list using recursion

SOLUTION:

1.

INPUT:

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

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

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

void linkedlisttraversal(struct node *ptr)
{
    printf("Linked list : ");
    while(ptr != NULL)
    {
        printf("%d ",ptr -> data);
        ptr = ptr -> next;
    }
    printf("\n");
}

struct node *insertatend(struct node *head,int data)
{
    struct node *ptr = (struct node *)malloc(sizeof(struct node));
    ptr -> data = data;
    struct node *p = head;

    while(p->next!=NULL)
    {
        p = p -> next;
    }
    p -> next = ptr;
    ptr -> next = NULL;
    return head;
}
int main()
```

```
{
    time_t t = time(NULL);
    struct tm tm = *localtime(&t);
    printf("Ram Krishna BT23CSE026\n");
    printf("%d-%02d-%02d %02d:%02d:%02d\n", tm.tm_year + 1900, tm.tm_mon + 1,
tm.tm_mday, tm.tm_hour, tm.tm_min, tm.tm_sec);
    printf("*********************************\n");
    struct node *head;
    struct node *first;
    struct node *second;
    struct node *third;

    head = (struct node *)malloc(sizeof(struct node));
    first = (struct node *)malloc(sizeof(struct node));
    second = (struct node *)malloc(sizeof(struct node));
    third = (struct node *)malloc(sizeof(struct node));

    head -> data = 7;
    head -> next = first;

    first -> data = 9;
    first -> next = second;

    second -> data = 12;
    second -> next = third;

    third -> data = 25;
    third -> next = NULL;

    printf("Linked list before insertion\n");
    linkedlisttraversal(head);
    head = insertatend(head,50);
    printf("Linked list after insertion\n");
    linkedlisttraversal(head);
```

```
    return 0;
}
```

OUTPUT:

```
PROBLEMS    OUTPUT    TERMINAL    DEBUG CONSOLE    PORTS                                                    Code

PS C:\Users\MANSI\Dropbox\BT23CSE026> cd "c:\Users\MANSI\Dropbox\BT23CSE026\DSA LAB\" ; if ($?) { gcc LAB4.c -o LAB4 } ; if ($?) { .\LAB4 }
Ram Krishna BT23CSE026
2024-02-18 23:27:29
**********************************
Linked list before insertion
Linked list : 7 9 12 25
Linked list after insertion
Linked list : 7 9 12 25 50
PS C:\Users\MANSI\Dropbox\BT23CSE026\DSA LAB>
```

2.

INPUT:

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

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

void linkedlistraversal(struct node *ptr)
{
    printf("Linked list : ");
    while(ptr != NULL)
    {
        printf("%d ",ptr -> data);
        ptr = ptr -> next;
    }
    printf("\n");
}

struct node * insertatbeginning(struct node *head, int data)
{
    struct node * ptr = (struct node *) malloc(sizeof(struct node));
    ptr->data = data;
    ptr->next = head;
```

```
    return ptr;
}
int main()
{
    time_t t = time(NULL);
    struct tm tm = *localtime(&t);
    printf("Ram Krishna BT23CSE026\n");
    printf("%d-%02d-%02d %02d:%02d:%02d\n", tm.tm_year + 1900, tm.tm_mon + 1,
tm.tm_mday, tm.tm_hour, tm.tm_min, tm.tm_sec);
    printf("********************************\n");
    struct node *head;
    struct node *first;
    struct node *second;
    struct node *third;

    head = (struct node *)malloc(sizeof(struct node));
    first = (struct node *)malloc(sizeof(struct node));
    second= (struct node *)malloc(sizeof(struct node));
    third= (struct node *)malloc(sizeof(struct node));

    head -> data = 7;
    head -> next = first;

    first -> data = 24;
    first -> next = second;

    second -> data = 53;
    second -> next = third;

    third -> data = 64;
    third-> next = NULL;

    printf("Linke list before insertion : \n");
    linkedlistraversal(head);
    head = insertatbeginning(head,95);
```

```
    printf("Linked list after insertion : \n");
    linkedlistraversal(head);
    return 0;
}
```

OUTPUT:

```
PROBLEMS    OUTPUT    TERMINAL    DEBUG CONSOLE    PORTS

PS C:\Users\MANSI\Dropbox\BT23CSE026> cd "c:\Users\MANSI\Dropbox\BT23CSE026\DSA LAB\" ; if ($?) { gcc LAB4.c -o LAB4 } ; if ($?)
Ram Krishna BT23CSE026
2024-02-18 23:28:17
*********************************
Linke list before insertion :
Linked list : 7 24 53 64
Linked list after insertion :
Linked list : 95 7 24 53 64
PS C:\Users\MANSI\Dropbox\BT23CSE026\DSA LAB>
```

3.

INPUT:

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

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

void linkedlistraversal(struct node *ptr)
{
    printf("Linked list : ");
    while(ptr != NULL)
    {
        printf("%d ",ptr -> data);
        ptr = ptr -> next;
    }
    printf("\n");
}

void printReverse(struct node* head)
```

```
{
    if (head == NULL)
    {
        return;
    }
    else
    {
    printReverse(head->next);
    printf("%d ", head->data);
    }
}

int main()
{
    time_t t = time(NULL);
    struct tm tm = *localtime(&t);
    printf("Ram Krishna BT23CSE026\n");
    printf("%d-%02d-%02d %02d:%02d:%02d\n", tm.tm_year + 1900, tm.tm_mon + 1,
tm.tm_mday, tm.tm_hour, tm.tm_min, tm.tm_sec);
    printf("********************************\n");
    struct node *head;
    struct node *first;
    struct node *second;
    struct node *third;
    struct node *fourth;

    head = (struct node *)malloc(sizeof(struct node));
    first= (struct node *)malloc(sizeof(struct node));
    second = (struct node *)malloc(sizeof(struct node));
    third = (struct node *)malloc(sizeof(struct node));
    fourth = (struct node *)malloc(sizeof(struct node));

    head -> data = 23;
    head -> next = first;
```

```
    first -> data = 33;
    first -> next = second;

    second -> data = 47;
    second -> next = third;

    third-> data = 15;
    third-> next = fourth;

    fourth -> data = 7;
    fourth -> next = NULL;

    printf("Linked list before reverse the nodes : \n");
    linkedlistraversal(head);
    printf("Linked list after reverse the nodes : \n");
    printReverse(head);
    return 0;
}
```

OUTPUT:

```
PROBLEMS    OUTPUT    TERMINAL    DEBUG CONSOLE    PORTS

PS C:\Users\MANSI\Dropbox\BT23CSE026> cd "c:\Users\MANSI\Dropbox\BT23CSE026\DSA LAB\" ; if ($?) { gcc LAB4.c -o LAB4 } ; if ($?)
Ram Krishna BT23CSE026
2024-02-18 23:29:02
********************************
Linked list before reverse the nodes :
Linked list : 23 33 47 15 7
Linked list after reverse the nodes :
7 15 47 33 23
PS C:\Users\MANSI\Dropbox\BT23CSE026\DSA LAB>
```

4.

INPUT:

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

struct node
{
    int data;
    struct node *next;
```

```
};

void linkedlisttraversal(struct node *ptr)
{
    printf("Linked list : ");
    while(ptr != NULL)
    {
        printf("%d ",ptr -> data);
        ptr = ptr -> next;
    }
    printf("\n");
}

int countnodes(struct node *head)
{
    int count = 0;
    struct node *current = head;
    while(current != NULL)
    {
        count ++;
        current = current -> next;
    }
    return count;
}

int main()
{
    time_t t = time(NULL);
    struct tm tm = *localtime(&t);
    printf("Ram Krishna BT23CSE026\n");
    printf("%d-%02d-%02d %02d:%02d:%02d\n", tm.tm_year + 1900, tm.tm_mon + 1,
tm.tm_mday, tm.tm_hour, tm.tm_min, tm.tm_sec);
    printf("**********************************\n");
    struct node *head;
    struct node *first;
```

```
    struct node *second;
    struct node *third;
    struct node *fourth;

    head = (struct node *)malloc(sizeof(struct node));
    first = (struct node *)malloc(sizeof(struct node));
    second = (struct node *)malloc(sizeof(struct node));
    third = (struct node *)malloc(sizeof(struct node));
    fourth = (struct node *)malloc(sizeof(struct node));

    head -> data = 7;
    head -> next = first;

    first -> data = 45;
    first -> next = second;

    second -> data = 67;
    second -> next = third;

    third -> data = 95;
    third -> next = fourth;

    fourth -> data = 81;
    fourth -> next = NULL;

    linkedlisttraversal(head);

    int nodecount = countnodes(head);
    printf("No of nodes in the linked list is ",nodecount);
    return 0;
}
```

OUTPUT:

```
PROBLEMS    OUTPUT    TERMINAL    DEBUG CONSOLE    PORTS

PS C:\Users\MANSI\Dropbox\BT23CSE026> cd "c:\Users\MANSI\Dropbox\BT23CSE026\DSA LAB\" ; if ($?) { gcc LAB4.c -o LAB4 }
Ram Krishna BT23CSE026
2024-02-18 23:29:45
********************************
Linked list : 7 45 67 95 81
No of nodes in the linked list is
PS C:\Users\MANSI\Dropbox\BT23CSE026\DSA LAB>
```

5.

INPUT:

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
struct Node
{
    int data;
    struct Node *next;
};
struct Node *convertRev(struct Node *head)
{
    if (head != NULL)
    {
        struct Node *previousNode = head;
        struct Node *currentNode = head->next;
        head = head->next;
        previousNode->next = NULL;
        while (head != NULL)
        {
            head = head->next;
            currentNode->next = previousNode;
            previousNode = currentNode;
            currentNode = head;
        }
        head = previousNode;
    }
    return head;
}
```

```
struct Node *addNode(struct Node *top, int data)
{
    struct Node *newNode = (struct Node *)malloc(sizeof(struct Node));
    newNode->data = data;
    newNode->next = NULL;
    if (top == NULL)
    {
        return newNode;
    }
    struct Node *p = top;
    while (p->next != NULL)
    {
        p = p->next;
    }
    p->next = newNode;
    return top;
}
int main()
{
    time_t t = time(NULL);
    struct tm tm = *localtime(&t);
    printf("Ram Krishna BT23CSE026\n");
    printf("%d-%02d-%02d %02d:%02d:%02d\n", tm.tm_year + 1900, tm.tm_mon + 1,
tm.tm_mday,
            tm.tm_hour, tm.tm_min, tm.tm_sec);
    printf("********************************\n");
    struct Node *head = NULL;
    head = addNode(head, 23);
    head = addNode(head, 3);
    head = addNode(head, 43);
    head = addNode(head, 10);
    struct Node *q = head;
    while (q != NULL)
    {
        printf("%d\t", q->data);
        q = q->next;
```

```
    }
    printf("\n");
    head = convertRev(head);
    struct Node *p = head;
    printf("After reversing the linked list:\n");
    while (p != NULL)
    {
        printf("%d\t", p->data);
        p = p->next;
    }
    return 0;
}
```

OUTPUT:

```
PS C:\Users\MANSI\Dropbox\CODES\DSA> cd "c:\Users\MANSI\Dropbox\COD
Ram Krishna BT23CSE026
2024-04-22 00:53:46
*********************************
23      3       43      10
After reversing the linked list:
10      43      3       23
PS C:\Users\MANSI\Dropbox\CODES\DSA\DSA LAB>
```

LAB 5
26/02/2024

QUESTION:

1. Create a linked list and delete the nodes with odd data and add nodes having even data.

2.In an array containing positive and negative integers,group the positive integers on one side and the negative on the other side,hence sort them in their group using merge sort for negative integers and quick sort for positive integers.

SOLUTION:

1.

INPUT:

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

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

void print(struct Node *ptr)
{
    if (ptr == NULL)
    {
        printf("Linked list does not exist");
        return;
    }
    while (ptr != NULL)
    {
        printf("Element: %d\n", ptr->data);
        ptr = ptr->next;
    }
}

struct Node *addNode(struct Node *top, int data)
{
    struct Node *newNode = (struct Node *)malloc(sizeof(struct Node));
    newNode->data = data;
    newNode->next = NULL;
    if (top == NULL)
    {
        return newNode;
```

```
    }

    struct Node *p = top;
    while (p->next != NULL)
    {
        p = p->next;
    }
    p->next = newNode;
    return top;
}

struct Node *deleteOddaddEven(struct Node *head)
{
    if (head == NULL)
    {
        printf("Empty linked list");
        return NULL;
    }
    while (head != NULL && head->data % 2 != 0)
    {
        struct Node *q = head;
        head = q->next;
        free(q);
    }
    struct Node *p = head;
    while (p != NULL)
    {
        if (p->data % 2 == 0)
        {
            struct Node *t = (struct Node *)malloc(sizeof(struct Node));
            t->data = p->data;
            t->next = p->next;
            p->next = t;
            p = t->next;
        }
```

```
        else
        {
            p = p->next;
        }
    }
    return head;
}

int main()
{
    time_t t = time(NULL);
    struct tm tm = *localtime(&t);
    printf("Ram Krishna BT23CSE026\n");
    printf("%d-%02d-%02d %02d:%02d:%02d\n", tm.tm_year + 1900, tm.tm_mon + 1,
tm.tm_mday, tm.tm_hour, tm.tm_min, tm.tm_sec);
    printf("********************************\n");
    struct Node *head = NULL;
    head = addNode(head, 21);
    head = addNode(head, 3);
    head = addNode(head, 42);
    head = addNode(head, 10);
    printf("Before function:\n");
    print(head);
    head = deleteOddaddEven(head);
    printf("After function:\n");
    print(head);
    return 0;
}
```

OUTPUT:

```
Ram Krishna BT23CSE026
2024-04-21 23:09:01
********************************
Before function:
Element: 21
Element: 3
Element: 42
Element: 10
After function:
Element: 42
Element: 42
Element: 10
Element: 10
PS C:\Users\MANSI\Dropbox\CODES\DSA\DSA LAB>
```

2.

INPUT:

```
#include <stdio.h>
#include <time.h>
void merge_array(int arr[], int lb, int mid, int ub)
{
    int i = lb;
    int j = mid + 1;
    int k = lb;
    int farr[50] = {0};
    while (i <= mid && j <= ub)
    {
        if (arr[i] <= arr[j])
        {
            farr[k] = arr[i];
            i++;
        }
        else
        {
            farr[k] = arr[j];
            j++;
        }
```

```
        k++;
    }
    while (i <= mid)
    {
        farr[k] = arr[i];
        i++;
        k++;
    }
    while (j <= ub)
    {
        farr[k] = arr[j];
        j++;
        k++;
    }
    for (int i = lb; i <= ub; i++)
    {
        arr[i] = farr[i];
    }
}
void mergeSort(int arr[], int lb, int ub)
{
    if (lb < ub)
    {
        int mid = (lb + ub) / 2;
        mergeSort(arr, lb, mid);
        mergeSort(arr, mid + 1, ub);
        merge_array(arr, lb, mid, ub);
    }
}
int partition(int arr[], int lb, int ub)
{
    int pivot = arr[lb];
    int i = lb + 1;
    int j = ub;
    do
    {
```

```
        while (arr[i] <= pivot)
        {
            i++;
        }
        while (arr[j] > pivot)
        {
            j--;
        }
        if (i < j)
        {
            int temp = arr[i];
            arr[i] = arr[j];
            arr[j] = temp;
        }
    } while (i < j);
    if (i > j)
    {
        int temp = arr[lb];
        arr[lb] = arr[j];
        arr[j] = temp;
    }
    return j;
}
void quickSort(int arr[], int lb, int ub)
{
    if (lb < ub)
    {
        int pivotIndex = partition(arr, lb, ub);
        quickSort(arr, lb, pivotIndex - 1);
        quickSort(arr, pivotIndex + 1, ub);
    }
}
int main()
{
    time_t t = time(NULL);
    struct tm tm = *localtime(&t);
```

```
    printf("RAM KRISHNA BT23CSE023\n");
    printf("%d-%02d-%02d %02d:%02d:%02d\n", tm.tm_year + 1900, tm.tm_mon + 1,
tm.tm_mday,
           tm.tm_hour, tm.tm_min, tm.tm_sec);
    printf("********************************\n");
    int n;
    printf("Enter size:");
    scanf("%d", &n);
    int arr[n];
    printf("Enter elements:");
    for (int i = 0; i < n; i++)
    {
        scanf("%d", &arr[i]);
    }
    int k = 0, l = n - 1;
    for (int i = 0; i <= l; i++)
    {
        if (arr[i] > 0)
        {
            int temp = arr[i];
            arr[i] = arr[k];
            arr[k] = temp;
            k++;
        }
        if (arr[i] < 0)
        {
            int temp = arr[i];
            arr[i] = arr[l];
            arr[l] = temp;
            l--;
        }
    }
    k += 2;
    int arr1[k];
    int arr2[n - k];
    for (int i = 0; i < k; i++)
```

```
    {
        arr1[i] = arr[i];
    }
    for (int i = k; i < n; i++)
    {
        arr2[i - k] = arr[i];
    }
    mergeSort(arr2, 0, n - k - 1);
    quickSort(arr1, 0, k - 1);
    for (int i = 0; i < k; i++)
    {
        arr[i] = arr1[i];
    }
    for (int i = k; i < n; i++)
    {
        arr[i] = arr2[i - k];
    }
    printf("After function:\n");
    for (int i = 0; i < n; i++)
    {
        printf("%d ", arr[i]);
    }
    return 0;
}
```

OUTPUT:

```
PS C:\Users\MANSI\Dropbox\CODES\DSA> cd "c:\Users\MANSI\Dropbox\
RAM KRISHNA BT23CSE026
2024-04-22 01:01:29
********************************
Enter size:5
Enter elements:36
12
85
7
65
After function:
1 1 7 12 36
PS C:\Users\MANSI\Dropbox\CODES\DSA\DSA LAB>
```

LAB 6
18/03/2024

QUESTION:

1.Create a circular singly linked list and perform deletion,insertion operations.
2.Create a circular doubly linked list and perform deletion and insertion operations.
3.Implement a stack using array.
4.Implement a stack using Linked list.
5.C program to convert infix expression to postfix expression.
6.C program to evaluate a postfix expression.

SOLUTION:

1.

INPUT:

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


struct node
```

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

void linkedlisttraversal(struct node *head)
{
    struct node *ptr = head;
    do
    {
        printf("Element : %d \n", ptr->data);
        ptr = ptr->next;
    } while (ptr != head);
}

struct node *createnode(int data)
{
    struct node *newnode = (struct node *)malloc(sizeof(struct node));
    newnode->data = data;
    newnode->next = NULL;
    return newnode;
}

struct node *insertatbeg(struct node *head, int data)
{
    struct node *ptr = (struct node *)malloc(sizeof(struct node));
    ptr->data = data;
    struct node *p = head;
    while (p->next != head)
    {
        p = p->next;
    }
    p->next = ptr;
    ptr->next = head;
    head = ptr;
    return head;
```

```
}

struct node *insertatend(struct node *head, int data)
{
    struct node *ptr = (struct node *)malloc(sizeof(struct node));
    ptr->data = data;
    struct node *p = head;
    while (p->next != head)
    {
        p = p->next;
    }
    p->next = ptr;
    ptr->next = head;
    return head;
}

struct node *insertatindex(struct node *head, int data, int index)
{
    struct node *ptr = (struct node *)malloc(sizeof(struct node));
    struct node *p = head;
    int i = 0;
    while (i != index - 1)
    {
        p = p->next;
        i++;
    }
    ptr->data = data;
    ptr->next = p->next;
    p->next = ptr;
    return head;
}

void deleteNode(struct node *head, int key) {
  struct node *temp = head, *prev;

  if (temp != NULL && temp->data == key) {
```

```
    head = temp->next;
    free(temp);
    return;
  }

  while (temp->next != head && temp->next->data != key) {
    temp = temp->next;
  }

  if (temp->next->data == key) {
    prev = temp;
    temp = temp->next;
    prev->next = temp->next;
    free(temp);
  }
}

int main()
{
    time_t t = time(NULL);
    struct tm tm = *localtime(&t);
    printf("Ram Krishna BT23CSE026\n");
    printf("%d-%02d-%02d %02d:%02d:%02d\n", tm.tm_year + 1900, tm.tm_mon + 1,
tm.tm_mday, tm.tm_hour, tm.tm_min, tm.tm_sec);
    printf("********************************\n");
    struct node *head = createnode(45);
    struct node *first = createnode(21);
    struct node *second = createnode(23);
    struct node *third = createnode(76);

    head->next = first;
    first->next = second;
    second->next = third;
    third->next = head;
```

```
    printf("Circular linked list before insertion : \n");
    linkedlisttraversal(head);
    printf("\n");
    head = insertatbeg(head, 12);
    head = insertatend(head, 91);
    head = insertatindex(head, 75, 2);
    printf("Circular linked list after insertion : \n");
    linkedlisttraversal(head);
    // deletenode(&head,23); -> This is when double pointer one function is used that is
void deletenode(struct node **head ,int key)
    deleteNode(head,12); // -> This is when function is used by taking single pointer.
    // There is some issue for the deletion at beginning by using this single pointer so
need to check it.
    printf("Circular linked list after deletion : \n");
    linkedlisttraversal(head);
    return 0;
}
```

OUTPUT:

```
Ram Krishna BT23CSE026
2024-04-22 00:23:44
**********************************
Circular linked list before insertion :
Element : 45
Element : 21
Element : 23
Element : 76

Circular linked list after insertion :
Element : 12
Element : 45
Element : 75
Element : 21
Element : 23
Element : 76
Element : 91
Circular linked list after deletion :
Element : 13244912
Element : 13244864
Element : 13238464
Element : 13243496
Element : 13254064
```

2.

INPUT:

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
struct Node
{
    int data;
    struct Node *prev;
    struct Node *next;
};
void print(struct Node *head)
{
    struct Node *ptr = head;
```

```
    if (ptr == NULL)
        return;
    do
    {
        printf("Element: %d\n", ptr->data);
        ptr = ptr->next;
    } while (ptr != head);
}
struct Node *insertAtStart(struct Node *head, int data)
{
    struct Node *ptr = (struct Node *)malloc(sizeof(struct Node));
    ptr->data = data;
    if (head == NULL)
    {
        ptr->next = ptr->prev = ptr;
    }
    else
    {
        ptr->next = head;
        ptr->prev = head->prev;
        head->prev->next = ptr;
        head->prev = ptr;
    }
    return ptr;
}
struct Node *insertInBetween(struct Node *head, int data, int index)
{
    struct Node *ptr = (struct Node *)(malloc(sizeof(struct Node)));
    ptr->data = data;
    struct Node *p = head;
    int i = 0;
    while (i != index - 1 && p != NULL)
    {
        p = p->next;
        i++;
    }
```

```
    if (p == NULL)
        return head;
    ptr->next = p->next;
    if (p->next != NULL)
        p->next->prev = ptr;
    ptr->prev = p;
    p->next = ptr;
    return head;
}
struct Node *insertAtEnd(struct Node *head, int data)
{
    struct Node *ptr = (struct Node *)(malloc(sizeof(struct Node)));
    ptr->data = data;
    ptr->next = NULL;
    struct Node *p = head;
    if (p == NULL)
        return ptr;
    while (p->next != NULL)
    {
        p = p->next;
    }
    p->next = ptr;
    ptr->prev = p;
    return head;
}
struct Node *deleteAtStart(struct Node *head)
{
    if (head == NULL)
        return NULL;
    struct Node *temp = head;
    if (head->next == head)
    {
        free(head);
        return NULL;
    }
    head = head->next;
```

```
    head->prev = temp->prev;
    temp->prev->next = head;
    free(temp);
    return head;
}
struct Node *deleteInBetween(struct Node *head, int index)
{
    if (head == NULL)
        return NULL;
    struct Node *p = head;
    int i = 0;
    while (i != index && p->next != head)
    {
        p = p->next;
        i++;
    }
    if (p->next == head)
        return head;
    p->prev->next = p->next;
    p->next->prev = p->prev;
    free(p);
    return head;
}
struct Node *deleteAtEnd(struct Node *head)
{
    if (head == NULL)
        return NULL;
    struct Node *toDelete = head->prev;
    if (head == head->next)
    {
        free(head);
        return NULL;
    }
    else
    {
        toDelete->prev->next = head;
```

```
        head->prev = toDelete->prev;
        free(toDelete);
    }
    return head;
}
int main()
{
    time_t t = time(NULL);
    struct tm tm = *localtime(&t);
    printf("RAM KRISHNA BT23CSE026\n");
    printf("%d-%02d-%02d %02d:%02d:%02d\n", tm.tm_year + 1900, tm.tm_mon + 1,
tm.tm_mday,
           tm.tm_hour, tm.tm_min, tm.tm_sec);
    printf("********************************\n");
    struct Node *head = NULL;
    struct Node *second = NULL;
    struct Node *third = NULL;
    struct Node *fourth = NULL;
    head = (struct Node *)(malloc(sizeof(struct Node)));
    second = (struct Node *)(malloc(sizeof(struct Node)));
    third = (struct Node *)(malloc(sizeof(struct Node)));
    fourth = (struct Node *)(malloc(sizeof(struct Node)));
    head->data = 7;
    head->next = second;
    head->prev = fourth;
    second->data = 10;
    second->next = third;
    second->prev = head;
    third->data = 13;
    third->next = fourth;
    third->prev = second;
    fourth->data = 17;
    fourth->next = head;
    fourth->prev = third;
    printf("Before insertion:\n");
    print(head);
```

```
int choice;
printf("1.Insertion at start, 2.Insertion in between, 3.Insertion at end: ");
scanf("%d", &choice);
switch (choice)
{
case 1:
{
    int data;
    printf("Enter data to insert at start: ");
    scanf("%d", &data);
    head = insertAtStart(head, data);
    printf("After insertion at start:\n");
    print(head);
    break;
}
case 2:
{
    int data, index;
    printf("Enter data to insert: ");
    scanf("%d", &data);
    printf("Enter index for insertion: ");
    scanf("%d", &index);
    head = insertInBetween(head, data, index);
    printf("After insertion in between:\n");
    print(head);
    break;
}
case 3:
{
    int data;
    printf("Enter data to insert at end: ");
    scanf("%d", &data);
    head = insertAtEnd(head, data);
    printf("After insertion at end:\n");
    print(head);
    break;
```

```
}
default:
    break;
}
printf("\n\nBefore deletion:\n");
print(head);
printf("1.Deletion at start, 2.Deletion in between, 3.Deletion at end: ");
scanf("%d", &choice);
switch (choice)
{
case 1:
{
    head = deleteAtStart(head);
    printf("After deletion at start:\n");
    print(head);
    break;
}
case 2:
{
    int index;
    printf("Enter index for deletion: ");
    scanf("%d", &index);
    head = deleteInBetween(head, index);
    printf("After deletion in between:\n");
    print(head);
    break;
}
case 3:
{
    head = deleteAtEnd(head);
    printf("After deletion at end:\n");
    print(head);
    break;
}
default:
    break;
```

```
    }
    return 0;
}
```

OUTPUT:

```
PS C:\Users\MANSI\Dropbox\CODES\DSA> cd "c:\Users\MANSI\Dropbox\CO
RAM KRISHNA BT23CSE026
2024-04-22 01:03:27
********************************
Before insertion:
Element: 7
Element: 10
Element: 13
Element: 17
1.Insertion at start, 2.Insertion in between, 3.Insertion at end:
Enter data to insert: 37
Enter index for insertion: 2
After insertion in between:
Element: 7
Element: 10
Element: 37
Element: 13
Element: 17


Before deletion:
Element: 7
Element: 10
Element: 37
Element: 13
Element: 17
1.Deletion at start, 2.Deletion in between, 3.Deletion at end: 3
After deletion at end:
Element: 7
Element: 10
Element: 37
Element: 13
PS C:\Users\MANSI\Dropbox\CODES\DSA\DSA LAB>
```

3.

INPUT:

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

```c
#include <stdlib.h>
#include <time.h>
struct Stack
{
    int size;
    int top;
    int *arr;
};
int isEmpty(struct Stack *ptr)
{
    if (ptr->top == -1)
    {
        return 1;
    }
    return 0;
}
int isFull(struct Stack *ptr)
{
    if (ptr->top == ptr->size - 1)
    {
        return 1;
    }
    return 0;
}
void printStack(struct Stack *ptr)
{
    for (int i = ptr->top; i >= 0; i--)
    {
        printf("%d\t", ptr->arr[i]);
    }
}
void push(struct Stack *ptr, int data)
{
    if (isFull(ptr))
    {
        printf("Stack Overflow...");
```

```
        return;
    }
    else
    {
        ptr->top++;
        ptr->arr[ptr->top] = data;
        printf("%d is pushed\n", data);
    }
}
void pop(struct Stack *ptr)
{
    if (isEmpty(ptr))
    {
        printf("Stack Underflow...");
    }
    else
    {
        int data = ptr->arr[ptr->top];
        ptr->top--;
        printf("\n%d is popped\n", data);
    }
}
int peek(struct Stack *ptr, int i)
{
    // i=>starting from top of stack
    if (ptr->top - i + 1 < 0)
    {
        printf("Invalid index");
        return -1;
    }
    else
    {
        return ptr->arr[ptr->top - i + 1];
    }
}
int main()
```

```
{
    time_t t = time(NULL);
    struct tm tm = *localtime(&t);
    printf("RAM KRISHNA BT23CSE026\n");
    printf("%d-%02d-%02d %02d:%02d:%02d\n", tm.tm_year + 1900, tm.tm_mon + 1,
tm.tm_mday,
            tm.tm_hour, tm.tm_min, tm.tm_sec);
    printf("********************************\n");
    struct Stack *s = (struct Stack *)malloc(sizeof(struct Stack));
    s->size = 8;
    s->top = -1;
    s->arr = (int *)malloc(s->size * sizeof(int));
    push(s, 100);
    push(s, 200);
    push(s, 300);
    push(s, 400);
    push(s, 500);
    printf("After pushing:\n");
    printStack(s);
    pop(s);
    printf("After popping:\n");
    printStack(s);
    printf("\nPeeked value= %d", peek(s, 3));
    free(s->arr);
    free(s);
    return 0;
}
```

OUTPUT:

```
PS C:\Users\MANSI\Dropbox\CODES\DSA> cd "c:\Users\MANS
RAM KRISHNA BT23CSE026
2024-04-22 01:10:27
********************************
100 is pushed
200 is pushed
300 is pushed
400 is pushed
500 is pushed
After pushing:
500     400     300     200     100
500 is popped
After popping:
400     300     200     100
Peeked value= 200
PS C:\Users\MANSI\Dropbox\CODES\DSA\DSA LAB>
```

4.

INPUT:

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
struct Node
{
    int data;
    struct Node *next;
};
struct Node *addNode(struct Node *top, int data)
{
    struct Node *newNode = (struct Node *)malloc(sizeof(struct Node));
    newNode->data = data;
    newNode->next = NULL;
    if (top == NULL)
    {
        return newNode;
    }
    struct Node *p = top;
```

```
    while (p->next != NULL)
    {
        p = p->next;
    }
    p->next = newNode;
    return top;
}
int isEmpty(struct Node *top)
{
    if (top == NULL)
    {
        return 1;
    }
    return 0;
}
int isFull(struct Node *n)
{
    if (n == NULL)
    {
        return 1;
    }
    return 0;
}
void print(struct Node *ptr)
{
    while (ptr != NULL)
    {
        printf("Element: %d\n", ptr->data);
        ptr = ptr->next;
    }
}
struct Node *pop(struct Node *top)
{
    if (isEmpty(top))
    {
        printf("Stack Underflow");
```

```
    }
    else
    {
        struct Node *p = top;
        top = top->next;
        free(p);
        return top;
    }
}
struct Node *push(struct Node *top, int data)
{
    struct Node *ptr = (struct Node *)malloc(sizeof(struct Node));
    if (isFull(ptr))
    {
        printf("Stack Overflow");
    }
    else
    {
        ptr->data = data;
        ptr->next = top;
        top = ptr;
        return top;
    }
}
int main()
{
    time_t t = time(NULL);
    struct tm tm = *localtime(&t);
    printf("RAM KRISHNA BT23CSE026\n");
    printf("%d-%02d-%02d %02d:%02d:%02d\n", tm.tm_year + 1900, tm.tm_mon + 1,
tm.tm_mday,
           tm.tm_hour, tm.tm_min, tm.tm_sec);
    printf("*********************************\n");
    struct Node *top = NULL;
    top = addNode(top, 23);
    top = addNode(top, 3);
```

```
    top = addNode(top, 43);
    top = addNode(top, 10);
    top = push(top, 100);
    // top=pop(top);
    print(top);
    return 0;
}
```

OUTPUT:

```
PS C:\Users\MANSI\Dropbox\CODES\DSA> cd "c:\Users\M
RAM KRISHNA BT23CSE026
2024-04-22 01:13:25
********************************
Element: 100
Element: 23
Element: 3
Element: 43
Element: 10
PS C:\Users\MANSI\Dropbox\CODES\DSA\DSA LAB>
```

5.

INPUT:

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <time.h>
struct Stack
{
    int size;
    int top;
    char *arr;
};
int isFull(struct Stack *ptr)
{
    return ptr->top == ptr->size - 1;
```

```
}
int isEmpty(struct Stack *ptr)
{
    return ptr->top == -1;
}
char pop(struct Stack *ptr)
{
    if (isEmpty(ptr))
    {
        printf("stack underflow");
        return '\0';
    }
    else
    {
        char poppedElement = ptr->arr[ptr->top];
        ptr->top--;
        return poppedElement;
    }
}
void push(struct Stack *ptr, char data)
{
    if (isFull(ptr))
    {
        printf("stack overflow");
    }
    else
    {
        ptr->top++;
        ptr->arr[ptr->top] = data;
    }
}
int precedence(char c)
{
    if (c == '^')
    {
        return 4;
```

```
    }
    else if (c == '*' || c == '/')
    {
        return 3;
    }
    else if (c == '+' || c == '-')
    {
        return 2;
    }
    else
    {
        return -1;
    }
}
int isOperator(char c)
{
    if (c == '+' || c == '-' || c == '*' || c == '/' || c == '^')
    {
        return 1;
    }
    else
    {
        return 0;
    }
}
char stackTop(struct Stack *s)
{
    return s->arr[s->top];
}
char *infixToPostfix(char *infix)
{
    struct Stack *s = (struct Stack *)malloc(sizeof(struct Stack));
    s->size = 100;
    s->top = -1;
    s->arr = (char *)malloc(s->size * sizeof(char));
    char *postfix = (char *)malloc((strlen(infix) + 1) * sizeof(char));
```

```
int i = 0; // infix index
int j = 0; // postfix index
while (infix[i] != '\0')
{
    if (!isOperator(infix[i]) && infix[i] != '(' && infix[i] != ')')
    {
        postfix[j] = infix[i];
        i++;
        j++;
    }
    else if (infix[i] == '(')
    {
        push(s, infix[i]);
        i++;
    }
    else if (infix[i] == ')')
    {
        while (!isEmpty(s) && stackTop(s) != '(')
        {
            postfix[j] = pop(s);
            j++;
        }
        if (!isEmpty(s) && stackTop(s) != '(')
        {
            printf("Mismatched parentheses");
            return '\0';
        }
        else
        {
            pop(s); // discard '('
        }
        i++;
    }
    else
    {
        while (!isEmpty(s) && precedence(infix[i]) <= precedence(stackTop(s)))
```

```
            {
                postfix[j] = pop(s);
                j++;
            }
            push(s, infix[i]);
            i++;
        }
    }
    while (!isEmpty(s))
    {
        postfix[j] = pop(s);
        j++;
    }
    postfix[j] = '\0';
    return postfix;
}
int main()
{
    time_t t = time(NULL);
    struct tm tm = *localtime(&t);
    printf("RAM KRISHNA BT23CSE026\n");
    printf("%d-%02d-%02d %02d:%02d:%02d\n", tm.tm_year + 1900, tm.tm_mon + 1,
tm.tm_mday,
           tm.tm_hour, tm.tm_min, tm.tm_sec);
    printf("*********************************\n");
    char *infix = "A+(B*C-(D/E^F)*G)*H/I/J-K";
    printf("Infix: %s\n", infix);
    char *postfix = infixToPostfix(infix);
    printf("Postfix expression: %s\n", postfix);
    return 0;
}
```

OUTPUT:

```
PS C:\Users\MANSI\Dropbox\CODES\DSA> cd "c:\Users\MANSI\Dropbox\CODES\DSA\DSA LAB\"
RAM KRISHNA BT23CSE026
2024-04-22 01:17:15
********************************
Infix: A+(B*C-(D/E^F)*G)*H/I/J-K
Postfix expression: ABC*DEF^/G*-H*I/J/+K-
PS C:\Users\MANSI\Dropbox\CODES\DSA\DSA LAB>
```

6.

INPUT:

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <time.h>
struct Stack
{
    int size;
    int top;
    char *arr;
};
int isFull(struct Stack *ptr)
{
    return ptr->top == ptr->size - 1;
}
int isEmpty(struct Stack *ptr)
{
    return ptr->top == -1;
}
char pop(struct Stack *ptr)
{
    if (isEmpty(ptr))
    {
        printf("stack underflow");
        return '\0';
    }
```

```
    else
    {
        char poppedElement = ptr->arr[ptr->top];
        ptr->top--;
        return poppedElement;
    }
}
void push(struct Stack *ptr, char data)
{
    if (isFull(ptr))
    {
        printf("stack overflow");
    }
    else
    {
        ptr->top++;
        ptr->arr[ptr->top] = data;
    }
}
int isOperator(char c)
{
    if (c == '+' || c == '-' || c == '*' || c == '/')
    {
        return 1;
    }
    else
    {
        return 0;
    }
}
int evalPostfix(char *postfix)
{
    struct Stack *s = (struct Stack *)malloc(sizeof(struct Stack));
    s->size = 40;
    s->top = -1;
    s->arr = (char *)malloc(s->size * sizeof(char));
```

```
int i = 0, res = 0;
while (postfix[i] != '\0')
{
    if (postfix[i] >= '0' && postfix[i] <= '9')
    {
        int num = postfix[i] - '0';
        push(s, num);
    }
    else if (isOperator(postfix[i]))
    {
        int s1 = pop(s);
        int s2 = pop(s);
        switch (postfix[i])
        {
        case '+':
        {
            res = s1 + s2;
            break;
        }
        case '-':
        {
            res = s2 - s1;
            break;
        }
        case '/':
        {
            res = s2 / s1;
            break;
        }
        case '*':
        {
            res = s2 * s1;
            break;
        }
        default:
            break;
```

```
            }
            push(s, res);
        }
        i++;
    }
    res = pop(s);
    return res;
}
int main()
{
    time_t t = time(NULL);
    struct tm tm = *localtime(&t);
    printf("RAM KRISHNA BT23CSE026\n");
    printf("%d-%02d-%02d %02d:%02d:%02d\n", tm.tm_year + 1900, tm.tm_mon + 1,
tm.tm_mday,
            tm.tm_hour, tm.tm_min, tm.tm_sec);
    printf("*********************************\n");
    char *postfix = "31/2*8-9+";
    printf("Postfix: %s\n", postfix);
    int result = evalPostfix(postfix);
    printf("Result: %d\n", result);
    return 0;
}
```

OUTPUT:

```
PS C:\Users\MANSI\Dropbox\CODES\DSA> cd "c:\User
RAM KRISHNA BT23CSE026
2024-04-22 01:20:18
*********************************
Postfix: 31/2*8-9+
Result: 7
PS C:\Users\MANSI\Dropbox\CODES\DSA\DSA LAB>
```

LAB 7
01/04/2024

QUESTION:

1.Sort the numbers in the stack given using a temporary stack.
2.Split a given circular linked list into two halves ,one having odd values and other having even values.

SOLUTION:

1.

INPUT:

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <time.h>
struct Stack {
int size;
int top;
int *arr;
};
int isFull(struct Stack *ptr) {
return ptr->top == ptr->size - 1;
}
int isEmpty(struct Stack *ptr) {
return ptr->top == -1;
}
int pop(struct Stack *ptr) {
if (isEmpty(ptr)) {
printf("Stack underflow");
return -1;
} else {
int poppedElement = ptr->arr[ptr->top];
ptr->top--;
return poppedElement;
}
```

```
}
void push(struct Stack *ptr, int data) {
if (isFull(ptr)) {
printf("Stack overflow");
} else {
ptr->top++;
ptr->arr[ptr->top] = data;
}
}
int stackTop(struct Stack *ptr) {
return ptr->arr[ptr->top];
}
void printStack(struct Stack *ptr) {
for (int i = ptr->top; i >= 0; i--) {
printf("%d ", ptr->arr[i]);
}
}
void sortStack(struct Stack *s) {
struct Stack *t = (struct Stack *)malloc(sizeof(struct Stack));
t->size = s->size;
t->top = -1;
t->arr = (int *)malloc(t->size * sizeof(int));
while (!isEmpty(s)) {
int temp = pop(s);
while (!isEmpty(t) && stackTop(t) < temp) {
push(s, pop(t));
}
push(t, temp);
}
while (!isEmpty(t)) {
push(s, pop(t));
}
free(t->arr);
free(t);
```

```
}
int main() {
    time_t t = time(NULL);
    struct tm tm = *localtime(&t);
    printf("RAM KRISHNA BT23CSE026\n");
    printf("%d-%02d-%02d %02d:%02d:%02d\n", tm.tm_year + 1900, tm.tm_mon + 1,
tm.tm_mday, tm.tm_hour, tm.tm_min, tm.tm_sec);
    printf("********************************\n");


struct Stack *s = (struct Stack *)malloc(sizeof(struct Stack));
s->size = 10;
s->top = -1;
s->arr = (int *)malloc(s->size * sizeof(int));
push(s, 1);
push(s, 5);
push(s, 5);
push(s, 2);
push(s, 3);
push(s, 8);
printf("Before sorting: ");
printStack(s);
sortStack(s);
printf("\nAfter sorting in descending order: ");
printStack(s);
printf("\n");
free(s->arr);
free(s);
return 0;
}
```

OUTPUT:

```
RAM KRISHNA BT23CSE026
2024-04-22 10:57:02
********************************
Before sorting: 8 3 2 5 5 1
After sorting in descending order: 8 5 5 3 2 1
PS C:\Users\Student\Desktop\ap>
```

2.

INPUT:

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


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


void print(struct Node *head) {
    if (head == NULL) return;
    struct Node *ptr = head;
    do {
        printf("Element: %d\n", ptr->data);
        ptr = ptr->next;
    } while (ptr != head);
}


struct Node *insertAtEnd(struct Node *head, int data) {
    struct Node* ptr = (struct Node*)malloc(sizeof(struct Node));
```

```
    ptr->data = data;


    if (head == NULL) {
        ptr->next = ptr;
        return ptr;
    }


    struct Node *p = head;
    while (p->next != head) {
        p = p->next;
    }
    p->next = ptr;
    ptr->next = head;
    return head;
}


void splitList(struct Node *head) {
    struct Node *head1 = NULL;
    struct Node *head2 = NULL;
    if (head == NULL) return;
    struct Node *p = head;
    do {
        if (p->data % 2 == 0) {
            head1 = insertAtEnd(head1, p->data);
        } else {
            head2 = insertAtEnd(head2, p->data);
        }
        p = p->next;
    } while (p != head);
    printf("Even elements:\n");
    print(head1);
```

```
    printf("Odd elements:\n");
    print(head2);
}


int main() {
    time_t t = time(NULL);
    struct tm tm = *localtime(&t);
    printf("RAM KRISHNA BT23CSE026\n");
    printf("%d-%02d-%02d %02d:%02d:%02d\n", tm.tm_year + 1900, tm.tm_mon + 1,
tm.tm_mday, tm.tm_hour, tm.tm_min, tm.tm_sec);
    printf("********************************\n");


    struct Node *head = NULL;
    struct Node *second = NULL;
    struct Node *third = NULL;
    struct Node *fourth = NULL;
    struct Node *fifth = NULL;
    struct Node *sixth = NULL;
    struct Node *seventh = NULL;
    head = (struct Node*)(malloc(sizeof(struct Node)));
    second = (struct Node*)(malloc(sizeof(struct Node)));
    third = (struct Node*)(malloc(sizeof(struct Node)));
    fourth = (struct Node*)(malloc(sizeof(struct Node)));
    fifth = (struct Node*)(malloc(sizeof(struct Node)));
    sixth = (struct Node*)(malloc(sizeof(struct Node)));
    seventh = (struct Node*)(malloc(sizeof(struct Node)));
    head->data = 1;
    head->next = second;
    second->data = 2;
    second->next = third;
    third->data = 3;
    third->next = fourth;
```

```
    fourth->data = 4;
    fourth->next = fifth;
    fifth->data = 5;
    fifth->next = sixth;
    sixth->data = 6;
    sixth->next = seventh;
    seventh->data = 7;
    seventh->next = head;
    splitList(head);
    return 0;
}
```

OUTPUT:

```
RAM KRISHNA BT23CSE026
2024-04-22 10:56:04
*********************************
Even elements:
Element: 2
Element: 4
Element: 6
Odd elements:
Element: 1
Element: 3
Element: 5
Element: 7
```

LAB 8
08/04/2024

QUESTION:

1.Count the leaf nodes in a tree.
2.Find the degree of a node of a tree.
3.Find the inorder, postorder and preorder traversal for a tree.

SOLUTION:

1.

INPUT:

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
typedef struct node
{
    struct node *right;
    struct node *left;
    int data;
} Node;
Node *createNode(int data)
{
    Node *t;
    t = (Node *)malloc(sizeof(Node));
    t->data = data;
    printf("Right Child of %d ?(y/n)", data);
    char ch;
    int n;
    scanf("%c", &ch);
    scanf("%c", &ch);
    if (ch == 'y' || ch == 'Y')
    {
        printf("Enter data : ");
        scanf("%d", &n);
        t->right = createNode(n);
    }
    else
    {
        t->right = NULL;
    }
    printf("Left Child of %d ?(y/n)", data);
    scanf("%c", &ch);
    scanf("%c", &ch);
```

```
    if (ch == 'y' || ch == 'Y')
    {
        printf("Enter data : ");
        scanf("%d", &n);
        t->left = createNode(n);
    }
    else
    {
        t->left = NULL;
    }
    return t;
}
int countLeafNodes(Node *root)
{
    if (root != NULL)
    {
        if (root->right == NULL && root->left == NULL)
        {
            return 1;
        }
        else
        {
            return countLeafNodes(root->right) + countLeafNodes(root->left);
        }
    }
}
int main()
{
    time_t t = time(NULL);
    struct tm tm = *localtime(&t);
    printf("RAM KRISHNA BT23CSE026\n");
    printf("%d-%02d-%02d %02d:%02d:%02d\n", tm.tm_year + 1900, tm.tm_mon + 1,
           tm.tm_mday, tm.tm_hour, tm.tm_min, tm.tm_sec);
    printf("********************************\n");
    int n;
    printf("Root data : ");
```

```
    scanf("%d", &n);
    Node *root = createNode(n);
    printf("\n");
    printf("Leaf Nodes = %d\n", countLeafNodes(root));
}
```

OUTPUT:

```
RAM KRISHNA BT23CSE026
2024-04-22 01:27:00
********************************
Root data : 5
Right Child of 5 ?(y/n)y
Enter data : 3
Right Child of 3 ?(y/n)n
Left Child of 3 ?(y/n)y
Enter data : 4
Right Child of 4 ?(y/n)n
Left Child of 4 ?(y/n)n
Left Child of 5 ?(y/n)y
Enter data : 2
Right Child of 2 ?(y/n)y
Enter data : 1
Right Child of 1 ?(y/n)n
Left Child of 1 ?(y/n)n
Left Child of 2 ?(y/n)y
Enter data : 0
Right Child of 0 ?(y/n)n
Left Child of 0 ?(y/n)n

Leaf Nodes = 3
PS C:\Users\MANSI\Dropbox\CODES\DSA\DSA LAB>
```

2.

INPUT:

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

```
typedef struct node
{
    struct node *right;
    struct node *left;
    int data;
} Node;
Node *createNode(int data)
{
    Node *t;
    t = (Node *)malloc(sizeof(Node));
    t->data = data;
    printf("Right Child of %d ?(y/n)", data);
    char ch;
    int n;
    scanf("%c", &ch);
    scanf("%c", &ch);
    if (ch == 'y' || ch == 'Y')
    {
        printf("Enter data : ");
        scanf("%d", &n);
        t->right = createNode(n);
    }
    else
    {
        t->right = NULL;
    }
    printf("Left Child of %d ?(y/n)", data);
    scanf("%c", &ch);
    scanf("%c", &ch);
    if (ch == 'y' || ch == 'Y')
    {
        printf("Enter data : ");
        scanf("%d", &n);
        t->left = createNode(n);
    }
    else
```

```
    {
        t->left = NULL;
    }
    return t;
}
int degreeOfNode(Node *root, int n)
{
    if (root != NULL)
    {
        int d = 0;
        if (root->data == n)
        {
            if (root->left != NULL)
                d++;
            if (root->right != NULL)
                d++;
        }
        else
        {
            d = degreeOfNode(root->left, n);
            if (d == 0)
                d = degreeOfNode(root->right, n);
        }
        return d;
    }
}
int main()
{
    time_t t = time(NULL);
    struct tm tm = *localtime(&t);
    printf("RAM KRISHNA BT23CSE026\n");
    printf("%d-%02d-%02d %02d:%02d:%02d\n", tm.tm_year + 1900, tm.tm_mon + 1,
           tm.tm_mday, tm.tm_hour, tm.tm_min, tm.tm_sec);
    printf("********************************\n");
    int n;
    printf("Root data : ");
```

```
    scanf("%d", &n);
    Node *root = createNode(n);
    printf("Enter a Node : ");
    scanf("%d", &n);
    printf("\n");
    printf("Degree of Node %d = %d\n", n, degreeOfNode(root, n));
}
```

OUTPUT:

```
RAM KRISHNA BT23CSE026
2024-04-22 01:30:32
*********************************
Root data : 5
Right Child of 5 ?(y/n)Y
Enter data : 3
Right Child of 3 ?(y/n)N
Left Child of 3 ?(y/n)Y
Enter data : 4
Right Child of 4 ?(y/n)N
Left Child of 4 ?(y/n)N
Left Child of 5 ?(y/n)Y
Enter data : 2
Right Child of 2 ?(y/n)Y
Enter data : 1
Right Child of 1 ?(y/n)N
Left Child of 1 ?(y/n)N
Left Child of 2 ?(y/n)Y
Enter data : 0
Right Child of 0 ?(y/n)N
Left Child of 0 ?(y/n)N
Enter a Node : 3

Degree of Node 3 = 1
PS C:\Users\MANSI\Dropbox\CODES\DSA\DSA LAB>
```

3.

INPUT:

```c
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
typedef struct node
{
    struct node *right;
    struct node *left;
    int data;
} Node;
void inorder(Node *root)
{
    if (root != NULL)
    {
        inorder(root->left);
        printf("%d ", root->data);
        inorder(root->right);
    }
}
void preorder(Node *root)
{
    if (root != NULL)
    {
        printf("%d ", root->data);
        preorder(root->left);
        preorder(root->right);
    }
}
void postorder(Node *root)
{
    if (root != NULL)
    {
        postorder(root->left);
        postorder(root->right);
        printf("%d ", root->data);
    }
```

```
}
int main()
{
    time_t t = time(NULL);
    struct tm tm = *localtime(&t);
    printf("RAM KRISHNA BT23CSE026\n");
    printf("%d-%02d-%02d %02d:%02d:%02d\n", tm.tm_year + 1900, tm.tm_mon + 1,
           tm.tm_mday, tm.tm_hour, tm.tm_min, tm.tm_sec);
    printf("**********************************\n");
    Node *first = (Node *)malloc(sizeof(Node));
    Node *second = (Node *)malloc(sizeof(Node));
    Node *third = (Node *)malloc(sizeof(Node));
    Node *fourth = (Node *)malloc(sizeof(Node));
    Node *fifth = (Node *)malloc(sizeof(Node));
    Node *sixth = (Node *)malloc(sizeof(Node));
    first->data = 1;
    second->data = 2;
    third->data = 3;
    fourth->data = 4;
    fifth->data = 5;
    sixth->data = 9;
    Node *root = first;
    first->left = second;
    first->right = third;
    second->left = fourth;
    second->right = fifth;
    third->left = NULL;
    third->right = sixth;
    fourth->left = NULL;
    fourth->right = NULL;
    fifth->left = NULL;
    fifth->right = NULL;
    sixth->left = NULL;
    sixth->right = NULL;
    printf("Inorder : ");
    inorder(root);
```

```
    printf("\n");
    printf("Preorder : ");
    preorder(root);
    printf("\n");
    printf("Postorder : ");
    postorder(root);
    printf("\n");
}
```

OUTPUT:

```
PROBLEMS    OUTPUT    TERMINAL    DEBUG CONSOLE    PORTS

PS C:\Users\MANSI\Dropbox\CODES\DSA> cd "c:\Users\MANSI\Dropl
RAM KRISHNA BT23CSE026
2024-04-22 01:34:56
********************************
Inorder : 4 2 5 1 3 9
Preorder : 1 2 4 5 3 9
Postorder : 4 5 2 9 3 1
PS C:\Users\MANSI\Dropbox\CODES\DSA\DSA LAB>
```

LAB 9
15/04/2024

QUESTION:

1.Insert in a binary search tree.

2.FInd minimum and maximum in a binary search tree.

SOLUTION:

1.

INPUT:

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

```c
#include <time.h>
typedef struct node
{
    struct node *left;
    int d;
    struct node *right;
} Node;
Node *insertInBST(Node *root, int d)
{
    if (root == NULL)
    {
        Node *newNode = (Node *)malloc(sizeof(Node));
        newNode->d = d;
        newNode->left = NULL;
        newNode->right = NULL;
        return newNode;
    }
    if (d < root->d)
    {
        root->left = insertInBST(root->left, d);
    }
    else
    {
        root->right = insertInBST(root->right, d);
    }
    return root;
}
void inorder(Node *root)
{
    if (root != NULL)
    {
        inorder(root->left);
        printf("%d ", root->d);
        inorder(root->right);
    }
}
```

```
int main()
{
    time_t t = time(NULL);
    struct tm tm = *localtime(&t);
    printf("RAM KRISHNA BT23CSE026\n");
    printf("%d-%02d-%02d %02d:%02d:%02d\n", tm.tm_year + 1900, tm.tm_mon + 1,
           tm.tm_mday, tm.tm_hour, tm.tm_min, tm.tm_sec);
    printf("********************************\n");
    int n;
    printf("Enter number of nodes in the BST (Assume numbering from extreme left of
each level) : ");
    scanf("%d", &n);
    Node *root = NULL;
    int d;
    printf("Enter data at node 1 : ");
    scanf("%d", &d);
    root = insertInBST(root, d);
    for (int x = 2; x <= n; x++)
    {
        printf("Enter data at node %d : ", x);
        scanf("%d", &d);
        root = insertInBST(root, d);
    }
    inorder(root);
    printf("\n");
    return 0;
}
```

OUTPUT:

```
 .\LAB8 }
RAM KRISHNA BT23CSE026
2024-04-22 01:40:47
*******************************
Enter number of nodes in the BST (Assume numbering from extr
Enter data at node 1 : 7
Enter data at node 2 : 6
Enter data at node 3 : 5
Enter data at node 4 : 4
Enter data at node 5 : 0
Enter data at node 6 : 1
0 1 4 5 6 7
PS C:\Users\MANSI\Dropbox\CODES\DSA\DSA LAB>
```

2.

OUTPUT:

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
typedef struct node
{
    struct node *left;
    int d;
    struct node *right;
} Node;
Node *insertInBST(Node *root, int d)
{
    if (root == NULL)
    {
        Node *newNode = (Node *)malloc(sizeof(Node));
        newNode->d = d;
        newNode->left = NULL;
        newNode->right = NULL;
        return newNode;
    }
    if (d < root->d)
```

```
    {
        root->left = insertInBST(root->left, d);
    }
    else
    {
        root->right = insertInBST(root->right, d);
    }
    return root;
}
void inorder(Node *root)
{
    if (root != NULL)
    {
        inorder(root->left);
        printf("%d ", root->d);
        inorder(root->right);
    }
}
int min(Node *root)
{
    if (root->left == NULL)
        return root->d;
    return min(root->left);
}
int max(Node *root)
{
    if (root->right == NULL)
        return root->d;
    return max(root->right);
}
int main()
{
    time_t t = time(NULL);
    struct tm tm = *localtime(&t);
    printf("RAM KRISHNA BT23CSE026\n");
    printf("%d-%02d-%02d %02d:%02d:%02d\n", tm.tm_year + 1900, tm.tm_mon + 1,
```

```
        tm.tm_mday, tm.tm_hour, tm.tm_min, tm.tm_sec);
    printf("*********************************\n");
    int n;
    printf("Enter number of nodes in the BST (Assume numbering from extreme left of
each level) : ");
    scanf("%d", &n);
    Node *root = NULL;
    int d;
    printf("Enter data at node 1 : ");
    scanf("%d", &d);
    root = insertInBST(root, d);
    for (int x = 2; x <= n; x++)
    {
        printf("Enter data at node %d : ", x);
        scanf("%d", &d);
        root = insertInBST(root, d);
    }
    inorder(root);
    printf("\n");
    printf("Min = %d\n", min(root));
    printf("Max = %d\n", max(root));
    return 0;
}
```

OUTPUT:

PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE PORTS

```
RAM KRISHNA BT23CSE026
2024-04-22 01:45:34
********************************
Enter number of nodes in the BST (Assume numbering fr
Enter data at node 1 : 3
Enter data at node 2 : 4
Enter data at node 3 : 2
Enter data at node 4 : 5
Enter data at node 5 : 0
Enter data at node 6 : 1
0 1 2 3 4 5
Min = 0
Max = 5
PS C:\Users\MANSI\Dropbox\CODES\DSA\DSA LAB>
```