

DATE:-22/09/22

COURSE NAME:-DATA STRUCTURES FOR EXPRESSION EVALUATION

COURSE CODE:-CSA0374

NAME OF THE STUDENT:- DILEEP.M

REGNO:-192111648

EXPERIMENT:-1(MATRIX OUT PUT)

The screenshot shows a C++ IDE with a file named 1ST.cpp. The code defines two 10x10 matrices, a and b, and a 10x10 result matrix mul. It prompts the user to enter the number of rows (r), the number of columns (c), and the first matrix element. It then prompts for the second matrix element. The output window shows the following text:

```
enter the number of row=2
enter the number of column=2
enter the first matrix element=
1
2
2
enter the second matrix element=
1
1
2
2
multiply of the matrix=
3 3
6 6
.....
Process exited after 18.43 seconds with return value 0
Press any key to continue . . .
```

EXPERIMENT:-2(EVEN ODD OUTPUT)

The screenshot shows a C++ IDE with a file named 1ST.cpp. The code defines two 10x10 matrices, a and b, and a 10x10 result matrix mul. It prompts the user to enter the number of rows (r), the number of columns (c), and the first matrix element. It then prompts for the second matrix element. The output window shows the following text:

```
enter the number of row=2
enter the number of column=2
enter the first matrix element=
1
2
2
enter the second matrix element=
1
1
2
2
multiply of the matrix=
3 3
6 6
.....
Process exited after 18.43 seconds with return value 0
Press any key to continue . . .
```

EXPERIMENT:-3(FACTORIAL WITHOUT OUTPUT)

The screenshot shows the Dev-C++ IDE with a project named 'factorial.cpp'. The code is as follows:

```
1 #include <stdio.h>
2 int main()
3 {
4     int n, i;
5     unsigned long long fact = 1;
6     printf("Enter an integer: ");
7     scanf("%d", &n);
8
9     // shows error if the user enters a negative integer
10    if (n < 0)
11        printf("Error! Factorial of a negative number doesn't exist.");
12    else
13    {
14        for (i = 1; i <= n; ++i) {
15            fact *= i;
16        }
17        printf("Factorial of %d = %llu", n, fact);
18    }
19    return 0;
20 }
```

The execution window shows the following output:

```
Enter an integer: 5
Factorial of 5 = 120
Process exited after 5.622 seconds with return value 0
Press any key to continue . . .
```

#### EXPERIMENT:4(FIBINOCCHI WITHOUT OUTPUT)

The screenshot shows the Dev-C++ IDE with a project named 'using fac.cpp'. The code is as follows:

```
1 #include<stdio.h>
2 int main()
3 {
4     int n1=0,n2=1,n3,i,number;
5     printf("Enter the number of elements:");
6     scanf("%d",&number);
7     printf("\n%d %d",n1,n2);//printing 0 and 1
8     for(i=2;i<=number;++i)//Loop starts from 2 because 0 and 1 are already printed
9     {
10        n3=n1+n2;
11        printf(" %d",n3);
12        n1=n2;
13        n2=n3;
14    }
15    return 0;
16 }
```

The execution window shows the following output:

```
Enter the number of elements:3
0 1 1
Process exited after 6.864 seconds with return value 0
Press any key to continue . . .
```

#### EXPERIMENT:5(FACTORIAL USING OUTPUT)

The screenshot shows the Dev-C++ IDE with a C++ program named 'using fac.cpp'. The code defines a recursive function 'multiplyNumbers' to calculate the factorial of a given integer 'n'. The main function prompts the user to enter a positive integer, reads the input, and calls the 'multiplyNumbers' function. The output window shows the execution result for an input of 8, resulting in a factorial of 40320.

```
#include<stdio.h>
long int multiplyNumbers(int n);
int main() {
    int n;
    printf("Enter a positive integer: ");
    scanf("%d",&n);
    printf("Factorial of %d = %ld", n, multiplyNumbers(n));
    return 0;
}

long int multiplyNumbers(int n) {
    if (n>=1)
        return n*multiplyNumbers(n-1);
    else
        return 1;
}
```

Compiler Output:

```
Enter a positive integer: 8
Factorial of 8 = 40320
Process exited after 3.288 seconds with return value 0
Press any key to continue . . .
```

## EXPERIMENT:6(FIBONOCCHI USING OUTPUT)

The screenshot shows the Dev-C++ IDE with a C++ program named 'using fac.cpp'. The code defines a recursive function 'printfibonacci' to generate the Fibonacci series up to 'n' elements. The main function prompts the user to enter the number of elements, reads the input, and calls the 'printfibonacci' function. The output window shows the execution result for an input of 10, resulting in the Fibonacci series: 0 1 1 2 3 5 8 13 21 34.

```
#include<stdio.h>
void printfibonacci(int n){
    static int n1=0,n2=1,n3;
    if(n>0){
        n3 = n1 + n2;
        n1 = n2;
        n2 = n3;
        printf("%d ",n3);
        printfibonacci(n-1);
    }
}

int main(){
    int n;
    printf("enter the number of elements: ");
    scanf("%d",&n);
    printf("fibonacci series: ");
    printf("%d %d ",0,1);
    printfibonacci(n-2); //n-2 because 2 numbers are already printed
    return 0;
}
```

Compiler Output:

```
enter the number of elements: 10
fibonacci series: 0 1 1 2 3 5 8 13 21 34
Process exited after 6.394 seconds with return value 0
Press any key to continue . . .
```

## EXPERIMENT:-7 (ARRAY OUTPUT)

The screenshot shows the Dev-C++ IDE with a C++ program for linear search. The code is as follows:

```
1 #include<stdio.h>
2 int findelement(int a[],int n,int key){
3     int i;
4     for( i=0;i<n;++i)
5     {
6         if (a[i]==key)
7             return i;
8     }
9     return -1;
10 }
11 int main(){
12     int a[]={3,5,7,9,8,22};
13     int n=sizeof(a)/sizeof(a[0]);
14     int key=9;
15     int position=findelement(a,n,key);
16     if(position== -1)
17     {
18         printf("elements %d notfound",key);
19     }
20     else
21     {
22         printf("position of %d:%d",key,position+9);
23     }
24     return 0;
25 }
26
```

The execution output window shows the following text:

```
C:\Users\chall\OneDrive\Documents\linear search.exe
position of 9:12
-----
Process exited after 0.06801 seconds with return value 0
Press any key to continue . . .
```

## ARRAY INSERTION(OUTPUT)

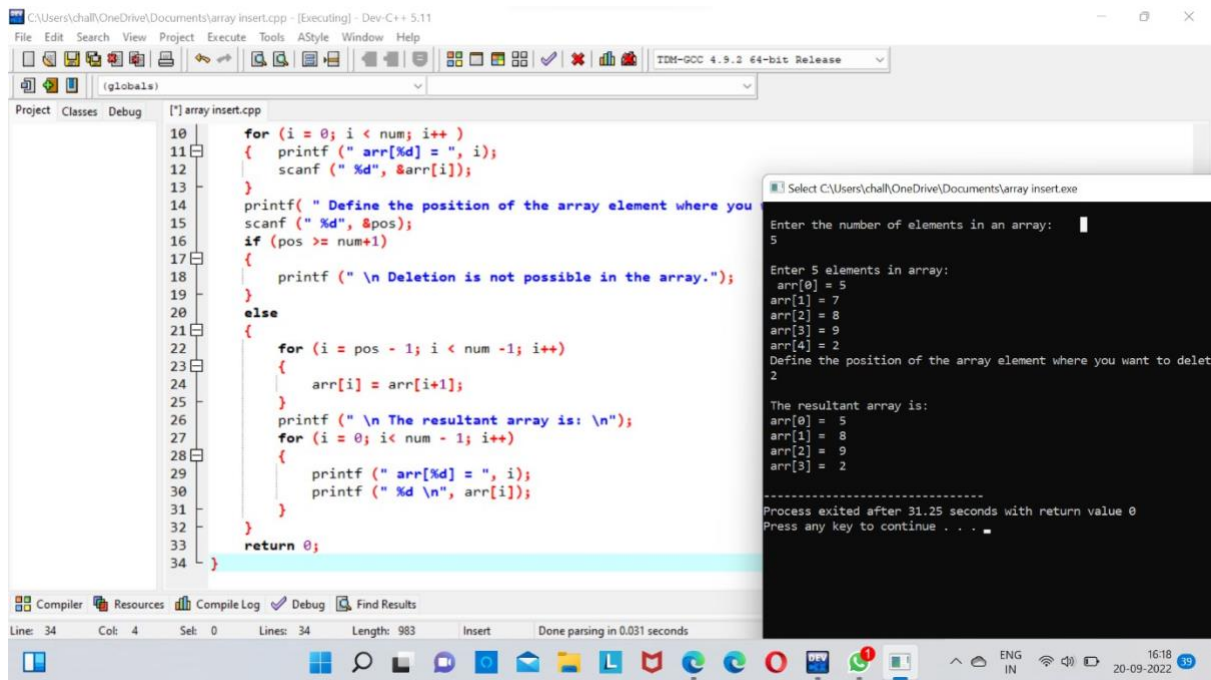
The screenshot shows the Dev-C++ IDE with a C++ program for array insertion. The code is as follows:

```
1 #include<stdio.h>
2 int main()
3 {
4     int position,c,n,value,array[50];
5     printf("enter the number if values in an array");
6     scanf("%d",&n);
7     printf("enter the values of %d\n",n);
8     for(c=0;c<n;c++)
9     {
10         scanf("%d",&array[c]);
11     }
12     printf("enter the location");
13     scanf("%d",&position);
14     printf("enter the value");
15     scanf("%d",&value);
16     for(c=n-1;c>=position-1;c--)
17     {
18         array[c+1]=array[c];
19     }
20     array[position-1]=value;
21     printf("resultant array is");
22     for(c=0;c<n;c++)
23     {
24         printf("%d ",array[c]);
25     }
26 }
```

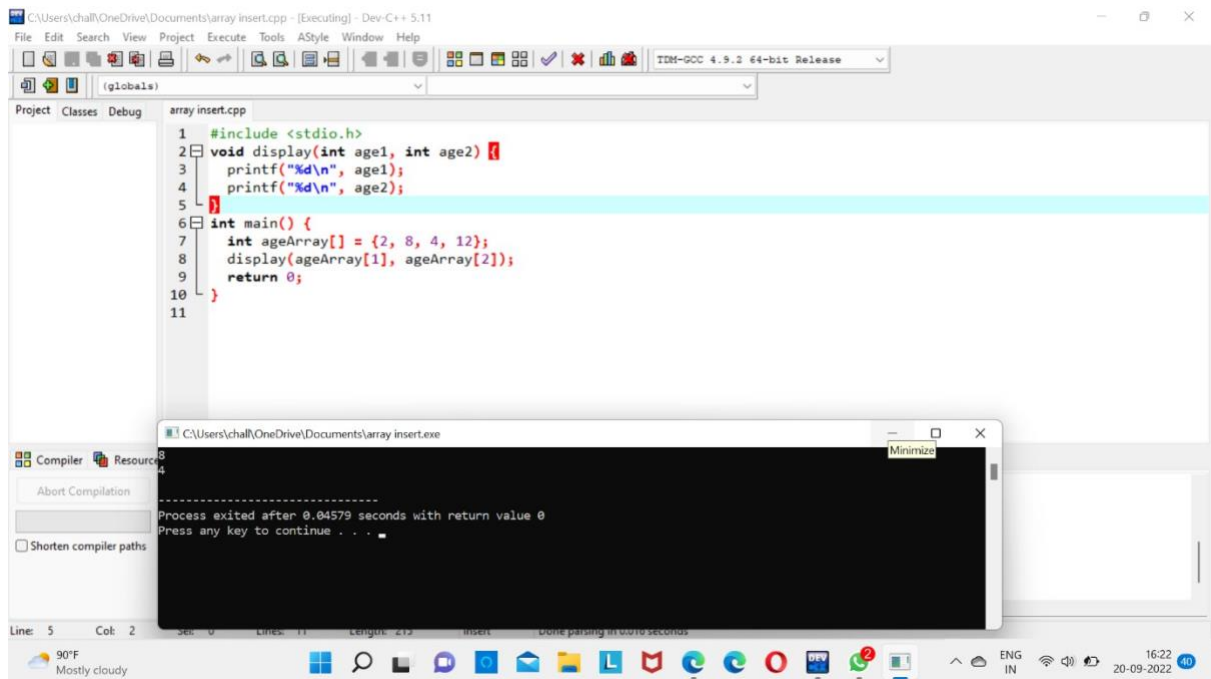
The execution output window shows the following text:

```
C:\Users\chall\OneDrive\Documents\array insert.exe
enter the number if values in an array3
enter the values of 3
5
8
9
enter the location2
enter the value9
resultant array is6
9
8
9
-----
Process exited after 18.26 seconds with return value 0
Press any key to continue . . .
```

## ARRAY DELETION(OUTPUT)



## ARRAY DISPLAY(OUTPUT)



## EXPERIMENT:-8(LINEARSEARCH(OUTPUT))



The screenshot shows the Dev-C++ IDE with a C++ program for linear search. The code defines a function `linearsearch` that iterates through an array `a` of size `n` to find a value `val`. The `main` function initializes an array `a = {3, 5, 7, 9, 8, 22}` and searches for `val = 21`. The output window shows the array elements, the search value, and the result: "elements is not present in the array".

```
1 #include<stdio.h>
2 int linearsearch(int a[],int n,int val){
3     for(int i=0;i<n;++i)
4     {
5         if (a[i]==val)
6             return i+1;
7     }
8     return -1;
9 }
10 int main(){
11     int a[]={3,5,7,9,8,22};
12     int val=21;
13     int n=sizeof(a)/sizeof(a[0]);
14     int res=linearsearch(a,n,val);
15     printf("the elements of the array are:");
16     for(int i=0;i<n;++i)
17         printf("%d",a[i]);
18     printf("\nelements to be searched is-%d",val);
19     if(res==-1)
20         printf("\nelements is not present in the array:");
21     else
22         printf("\nelements is presents at %d position of array",res);
23     return 0;
24 }
25
26
```

Output:

```
Select C:\Users\chall\OneDrive\Documents\linear search.exe
the elements of the array are:3579822
elements to be searched is-21
elements is not present in the array:
Process exited after 0.08195 seconds with return value 0
Press any key to continue . . .
```

## EXPERIMENT:9(BINARYSEARCH(OUTPUT))

The screenshot shows the Dev-C++ IDE with a C++ program for binary search. The code defines a function `binarySearch` that uses a while loop to find a value `x` in a sorted array. The `main` function initializes an array `array = {3, 4, 5, 6, 7, 8, 9}` and searches for `x = 4`. The output window shows the element found at index 1.

```
1 #include <stdio.h>
2 int binarySearch(int array[], int x, int low, int high) {
3     // Repeat until the pointers low and high meet each other
4     while (low <= high) {
5         int mid = low + (high - low) / 2;
6         if (array[mid] == x)
7             return mid;
8         if (array[mid] < x)
9             low = mid + 1;
10        else
11            high = mid - 1;
12    }
13    return -1;
14 }
15 int main(void) {
16     int array[] = {3, 4, 5, 6, 7, 8, 9};
17     int n = sizeof(array) / sizeof(array[0]);
18     int x = 4;
19     int result = binarySearch(array, x, 0, n - 1);
20     if (result == -1)
21         printf("Not found");
22     else
23         printf("Element is found at index %d", result);
24     return 0;
25 }
26
```

Output:

```
C:\Users\chall\OneDrive\Documents\linear search.exe
Element is found at index 1
Process exited after 0.06617 seconds with return value 0
Press any key to continue . . .
```

## EXPERIMENT:10(LINKED LIST(OUTPUT))

```
C:\Users\chall\OneDrive\Documents\array insert.cpp - Dev-C++ 5.11
File Edit Search View Project Execute Tools AStyle Window Help
(globals)
Project Classes Debug
[*] array insert.cpp
79 void printList(struct Node* node) {
80     while (node != NULL) {
81         printf(" %d ", node->data);
82         node = node->next;
83     }
84 }
85 int main() {
86     struct Node* head = NULL;
87     insertAtEnd(&head, 1);
88     insertAtBeginning(&head, 2);
89     insertAtBeginning(&head, 3);
90     insertAtEnd(&head, 4);
91     insertAfter(head->next, 5);
92     printf("Linked list: ");
93     printList(head);
94     printf("\nAfter deleting an element: ");
95     deleteNode(&head, 3);
96     printList(head);
97     int item_to_find = 3;
98     if (searchNode(&head, item_to_find)) {
99         printf("\n%d is found", item_to_find);
100     } else {
101         printf("\n%d is not found", item_to_find);
102     }
103     sortLinkedList(&head);
104     printf("\nSorted List: ");
105 }
Compiler Resources Compile Log Debug Find Results
Line: 79 Col: 3 Sel: 0 Lines: 106 Length: 2754 Insert Done parsing
90°F Mostly cloudy
```

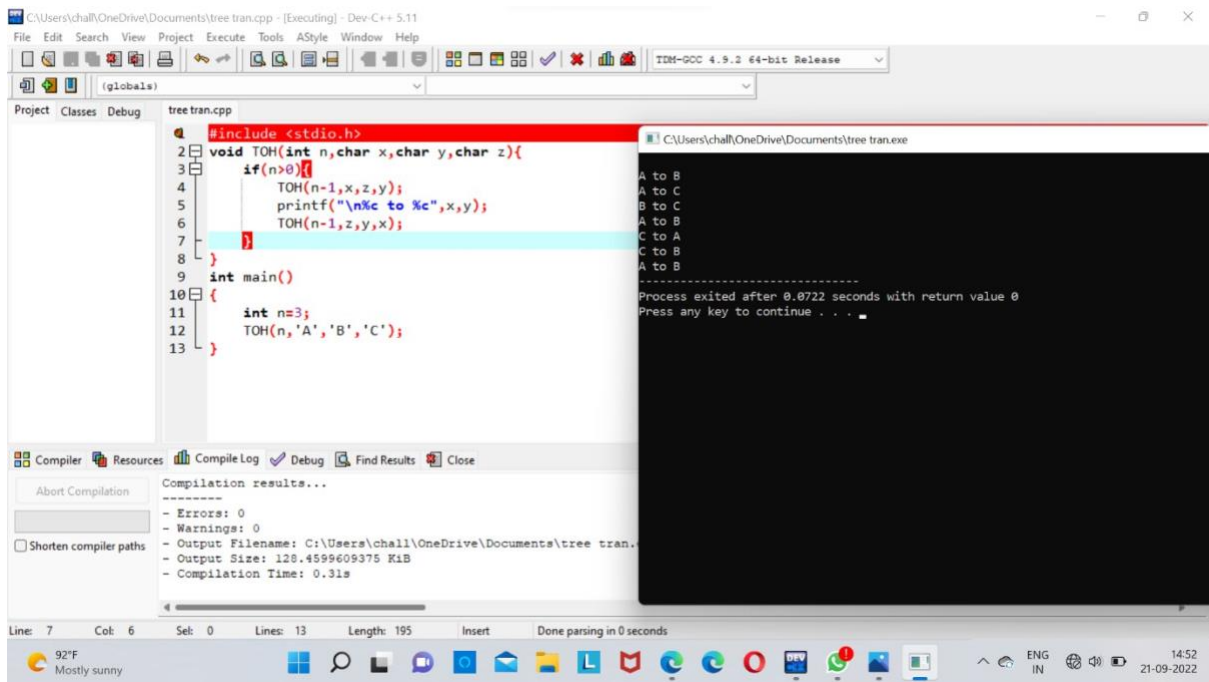
```
C:\Users\chall\OneDrive\Documents\array insert.exe
Linked list: 3 2 5 1 4
After deleting an element: 2 5 1 4
3 is not found
Sorted list: 1 2 4 5
-----
Process exited after 0.05659 seconds with return value 0
Press any key to continue . . .
```

## EXPERIMENT:11(STACK OPERATIONS(OUTPUT))

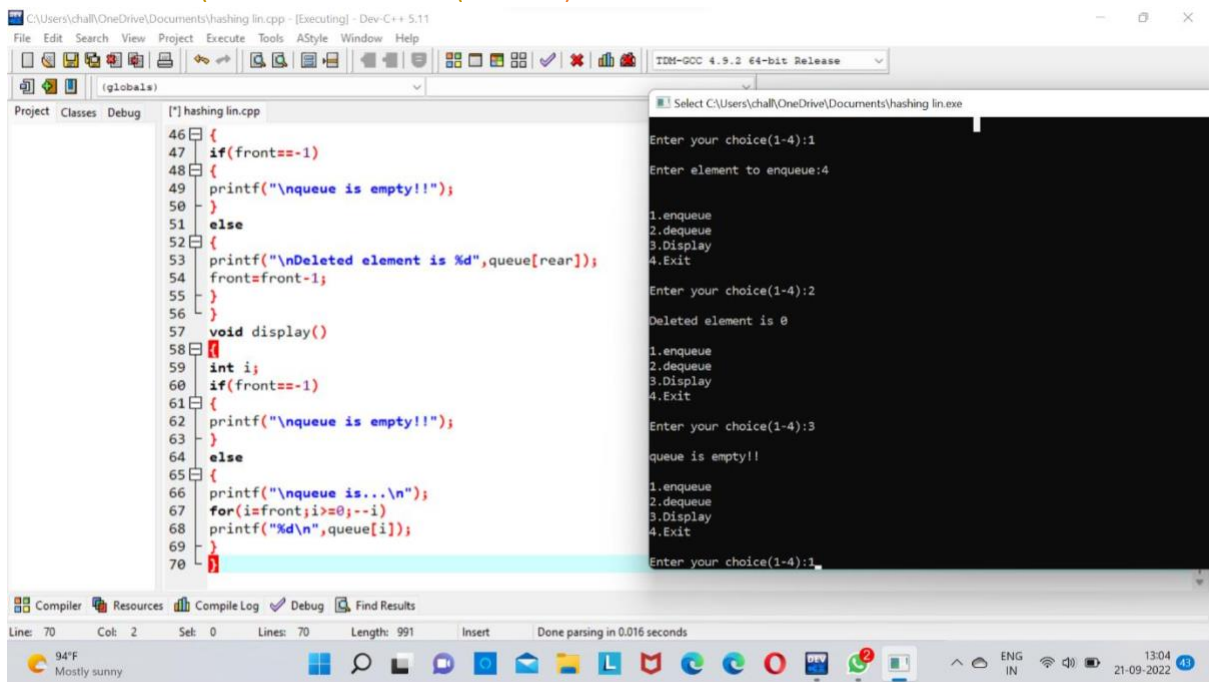
```
C:\Users\chall\OneDrive\Documents\tree tran.cpp - [Executing] - Dev-C++ 5.11
File Edit Search View Project Execute Tools AStyle Window Help
(globals)
Project Classes Debug
tree tran.cpp
1 #include <stdio.h>
2 void TOH(int n,char x,char y,char z){
3     if(n>0){
4         TOH(n-1,x,z,y);
5         printf("\n%c to %c",x,y);
6         TOH(n-1,z,y,x);
7     }
8 }
9 int main()
10 {
11     int n=3;
12     TOH(n,'A','B','C');
13 }
Compiler Resources Compile Log Debug Find Results Close
Compilation results...
- Errors: 0
- Warnings: 0
- Output Filename: C:\Users\chall\OneDrive\Documents\tree tran.
- Output Size: 128.4599609375 KiB
- Compilation Time: 0.31s
Line: 7 Col: 6 Sel: 0 Lines: 13 Length: 195 Insert Done parsing in 0 seconds
92°F Mostly sunny
```

```
C:\Users\chall\OneDrive\Documents\tree tran.exe
A to B
A to C
B to C
A to B
C to A
C to B
A to B
-----
Process exited after 0.0722 seconds with return value 0
Press any key to continue . . .
```

## EXPERIMENT:12(STACK APPLICATION)



### EXPERIMENT:13(QUEUE OPERATIONS(OUTPUT))



### EXPERIMENT:14(TREE TRANSVERSAL(OUTPUT))



```
tree tran.cpp
27 void printInOrder(struct node* node)
28 {
29     if (node == NULL)
30         return;
31     printInOrder(node->left);
32     printf("%d ", node->data);
33     printInOrder(node->right);
34 }
35 void printPreOrder(struct node* node)
36 {
37     if (node == NULL)
38         return;
39     printf("%d ", node->data);
40     printPreOrder(node->left);
41     printPreOrder(node->right);
42 }
43 int main()
44 {
45     struct node *root = newNode(1);
46     root->left = newNode(2);
47     root->right = newNode(3);
48     root->left->left = newNode(4);
49     root->left->right = newNode(5);
50
51     printf("\nPreorder traversal of binary tree is \n");
52     printPreOrder(root);
53 }
```

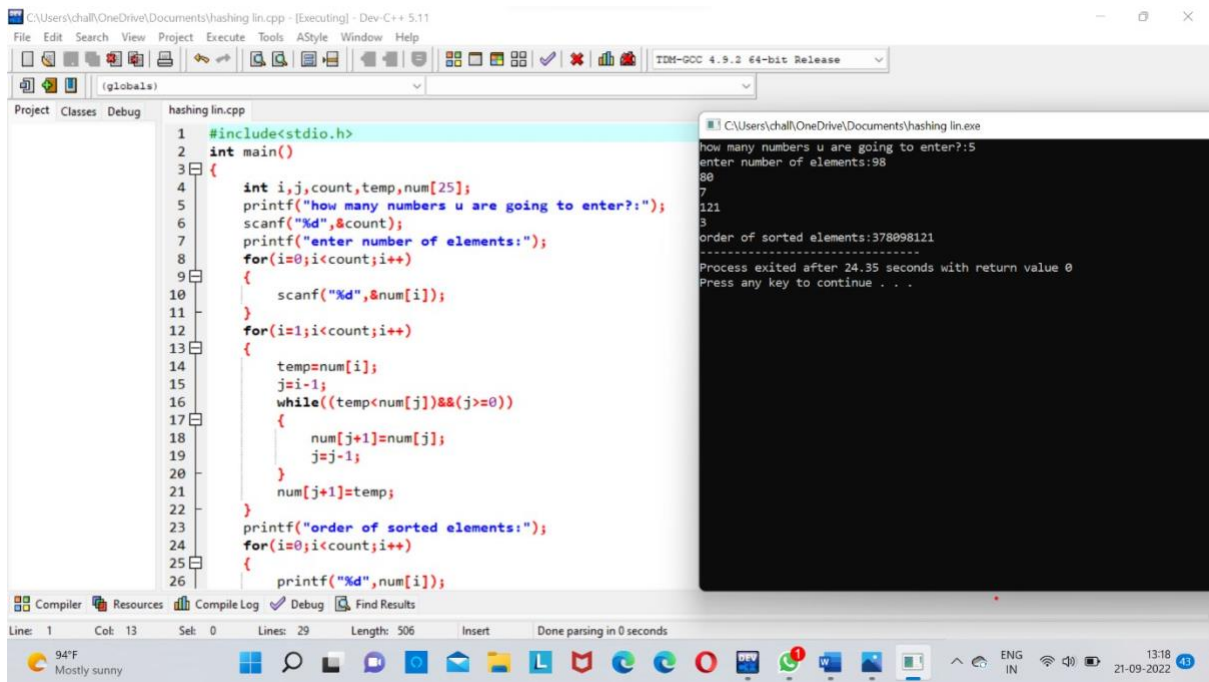
```
C:\Users\chall\OneDrive\Documents\tree tran.exe
Preorder traversal of binary tree is
1 2 4 5 3
Inorder traversal of binary tree is
4 2 5 1 3
Postorder traversal of binary tree is
4 5 2 3 1
```

## EXPERIMENT:-15(HASH USING LINEAR PROBING(OUTPUT))

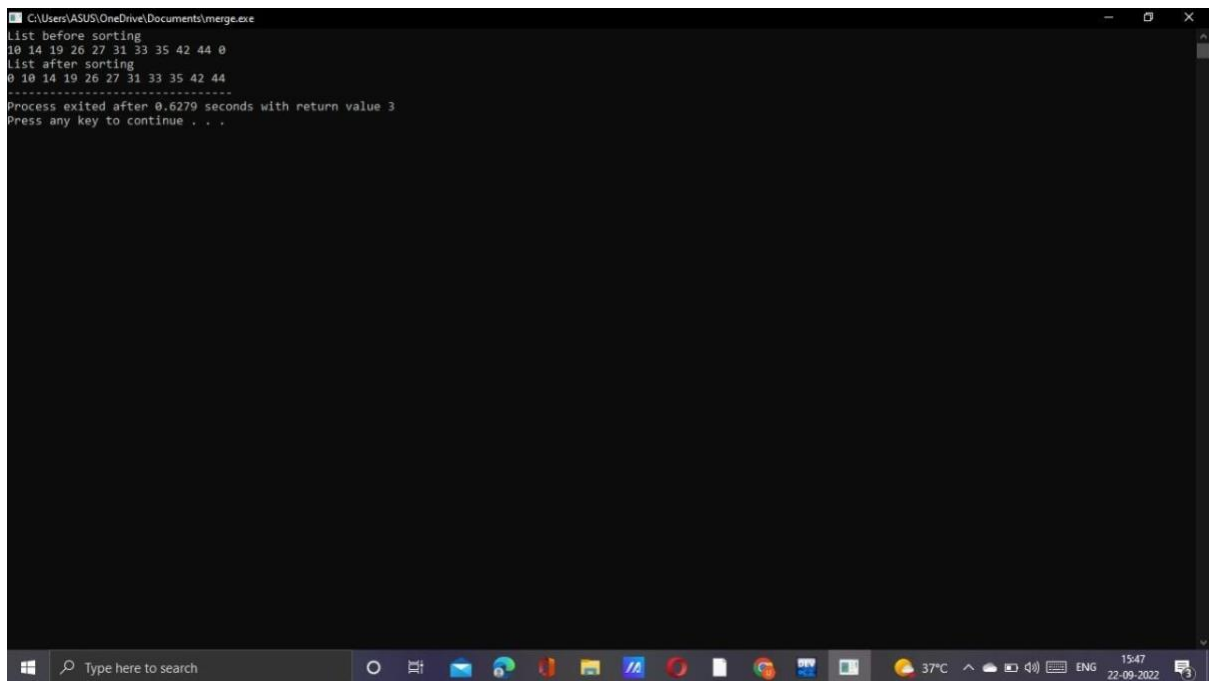
```
hashing lin.cpp
27 scanf("%d",&key);
28 hkey=key%TABLE_SIZE;
29 for(i=0;i<TABLE_SIZE; i++)
30 {
31     index=(hkey+i)%TABLE_SIZE;
32     if(h[index]==key)
33     {
34         printf("value is found at index %d",index);
35         break;
36     }
37 }
38 if(i == TABLE_SIZE)
39     printf("\n value is not found\n");
40 }
41 void display()
42 {
43     int i;
44     printf("\nelements in the hash table are \n");
45     for(i=0;i< TABLE_SIZE; ++i)
46         printf("\nat index %d \t value = %d",i,h[i]);
47 }
48 main()
49 {
50     int opt,i;
51     while(1)
52     {
```

```
Select C:\Users\chall\OneDrive\Documents\hashing lin.exe
1
enter a value to insert into hash table
22
Press 1. Insert 2. Display 3. Search 4.Exit
2
elements in the hash table are
at index 0 value = 0
at index 1 value = 0
at index 2 value = 22
at index 3 value = 0
at index 4 value = 0
at index 5 value = 0
at index 6 value = 0
at index 7 value = 0
at index 8 value = 8
at index 9 value = 0
Press 1. Insert 2. Display 3. Search 4.Exit
3
enter search element
42
value is not found
Press 1. Insert 2. Display 3. Search 4.Exit
```

## EXPERIMENT:16(INSERTION SORTING(OUTPUT))



## EXPERIMENT:17(MERGE SORTING)



## EXPERIMENT:-18(QUICK SORT(OUTPUT))

```
C:\Users\ASUS\OneDrive\Documents\Quick.exe
Enter total no.of elements: 7
Enter the elements: 10
5
25
48
68
98
100
Sorted Array: 5 10 25 28 48 68 100
-----
Process exited after 18.39 seconds with return value 0
Press any key to continue . . .
```

## EXPERIMENT:-19(HEAP SORT (OUTPUT))

```
C:\Users\ASUS\OneDrive\Documents\heap.exe
Before sorting array elements are -
48 10 23 43 26 1
After sorting array elements are -
1 10 23 26 28 43 48
-----
Process exited after 0.7099 seconds with return value 0
Press any key to continue . . .
```

## EXPERIMENT:20(AVL OPERATIONS(OUTPUT))

```
C:\Users\ASUS\OneDrive\Documents\AVL trees.exe
4 2 1 3 7 5 8
After deletion: 4 2 1 7 5 8
-----
Process exited after 0.61 seconds with return value 0
Press any key to continue . . .
```

## EXPERIMENT:21(BFS(OUTPUT))

```
C:\Users\ASUS\OneDrive\Documents\Breadth.exe

Enter the number of vertices:4

Enter graph data in matrix form:
1 1 1 1
0 1 0 0
0 0 1 0
0 0 0 1

Enter the starting vertex:1

The node which are reachable are:
1      2      3      4
-----
Process exited after 26.89 seconds with return value 0
Press any key to continue . . .
```

## EXPERIMENT:22(DFS(OUTPUT))

```
C:\Users\ASUS\OneDrive\Documents\Depth.exe
Enter no. of vertices:4
Enter the adjacency matrix:
1 1 1 1
1 0 1 0
0 0 0 0
0 0 0 1
Enter the starting node:1
Distance of node0=1
Path=0<-1
Distance of node2=1
Path=2<-1
Distance of node3=2
Path=3<-0<-1
-----
Process exited after 35.38 seconds with return value 0
Press any key to continue . . .
```

## EXPERIMENT:23(DIJKSTRA ALGORITHM(OUTPUT))

```
C:\Users\ASUS\OneDrive\Documents\Dijkstra's.exe
Distance from source to 1: 3
Distance from source to 2: 1
Distance from source to 3: 2
Distance from source to 4: 4
Distance from source to 5: 4
Distance from source to 6: 3
-----
Process exited after 0.5575 seconds with return value 0
Press any key to continue . . .
```

## EXPERIMENT:24(PRIM'S ALGORITHM(OUTPUT))



```
C:\Users\ASUS\OneDrive\Documents\Prim's.exe

Edge      Weight
3 <-> 1    4
0 <-> 2    3
2 <-> 3    2
3 <-> 4    1

-----
Process exited after 0.6292 seconds with return value 0
Press any key to continue . . .
```

## EXPERIMENT:25(KRUKAL ALGORITHM(OUTPUT)

```
C:\Users\ASUS\OneDrive\Documents\Kruskal.exe

2 - 1 : 2
5 - 2 : 2
3 - 2 : 3
4 - 3 : 3
1 - 0 : 4
Spanning tree cost: 14
-----
Process exited after 0.9241 seconds with return value 23
Press any key to continue . . .
```