



Graphic Era

HILL UNIVERSITY

Established by an Act of the State Legislature of Uttarakhand (Adhiniyam Sankhya 12 of 2011)
University under section 2(f) of UGC Act, 1956

Data Structures And File Organization End Term File

Submitted To: Mrs Manisha
Koranga

Submitted By: Deepankar
Sharma



NAME: DEEPAK SHARMA
COURSE: BCA-HALDWANI
UNIVERSITY ROLL NO: 20041299
STUDENT ID : 2091299

Data Structures And File

Organizations

TBC 201

INDEX

Sr. No.	Assign. Date	Subm. Date	<u>Title</u>	Teacher's Remark
1.		23 APRIL 2021	Lab Assignment 1	
2.		30 APRIL 2021	Lab Assignment 2	
3.		13 MAY 2021	Lab Assignment 3	
4.		13 MAY 2021	Lab Assignment 4	
5.		21 MAY 2021	Lab Assignment 5	
6.		26 MAY 2021	Lab Assignment 6	
7.		27 MAY 2021	Lab Assignment 7	
8.		03 August 2021	Lab Assignment 8	

9.		27 August 2021	Lab Assignment 9	
----	--	-------------------	------------------	--

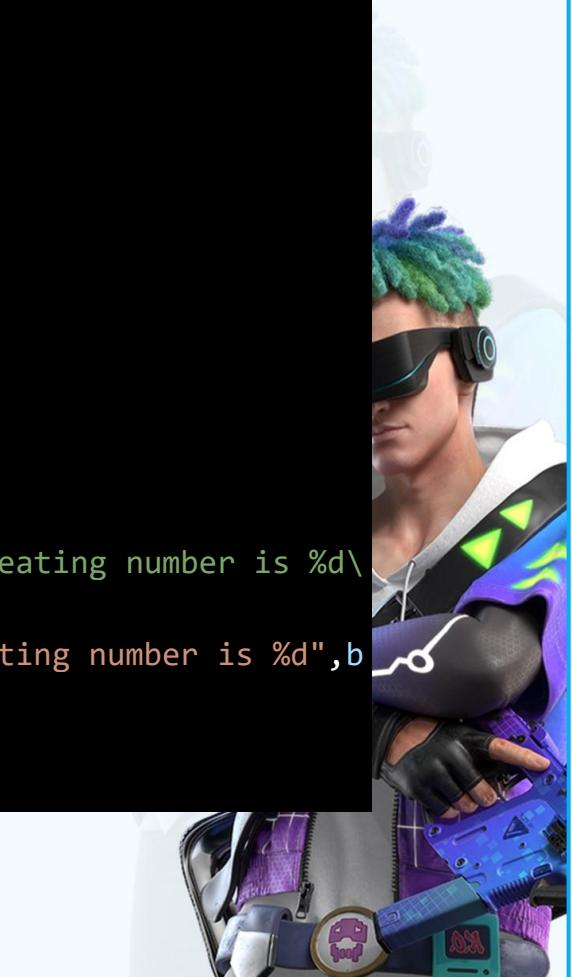


Lab Assignment 1

01_second_non_repeating_num_in_array

```
#include<stdio.h>
int main()
{
    int a[10],b[2],freq=0, count=0;
    printf("Enter the elements of the array : \n");
    for(int i=0; i<10; i++)
        scanf("%d",&a[i]);

    for(int i=0; i<10; i++)
    {
        freq=1;
        for(int j=0; j<10; j++)
        {
            if(a[i]==a[j]&& i!=j)
            {
                freq++;
            }
        }
        if(freq==1)
        {
            b[count]=a[i];
            count++;
        }
        if(count==2)break;
    }
    // printf("\n\nThe first non repeating number is %d\n",b[0]);
    printf("\n\nThe second non repeating number is %d",b[1]);
    return 0;
}
```



```

File Edit Selection View Go Run Terminal Help 01_second_non_repeating_num_in_array.cpp - DS-ALGO - Visual Studio Code
and_small_in_Array.cpp 05_occurrence_of_any_num_in_Array.cpp 01_multiplicationOfMatrices.cpp 01_second_non_repeating_num_in_array.cpp
EXPLORER OPEN EDITORS DS labAssignment1 > 01_second_non_repeating_num_in_array.cpp > main()
1 #include<stdio.h>
2 int main()
3 {
4     int a[10],b[2],freq=0, count=0;
5     printf("Enter the elements of the array : \n");
6     for(int i=0; i<10; i++)
7         scanf("%d",&a[i]);
8
9     for(int i=0; i<10; i++)
10    {
11        freq=1;
12        for(int j=0; j<10; j++)
13        {
14            if(a[i]==a[j]&& i!=j)
15            {
16                freq++;
17            }
18        }
19        if(freq==1)
20        {
21            b[0]=a[i];
22            b[1]=i;
23        }
24    }
25
26    if(b[0]==0)
27    {
28        printf("No second non repeating number found");
29    }
30    else
31    {
32        printf("The second non repeating number is %d",b[0]);
33    }
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
99

```

02_reverse_Array_using_swapping

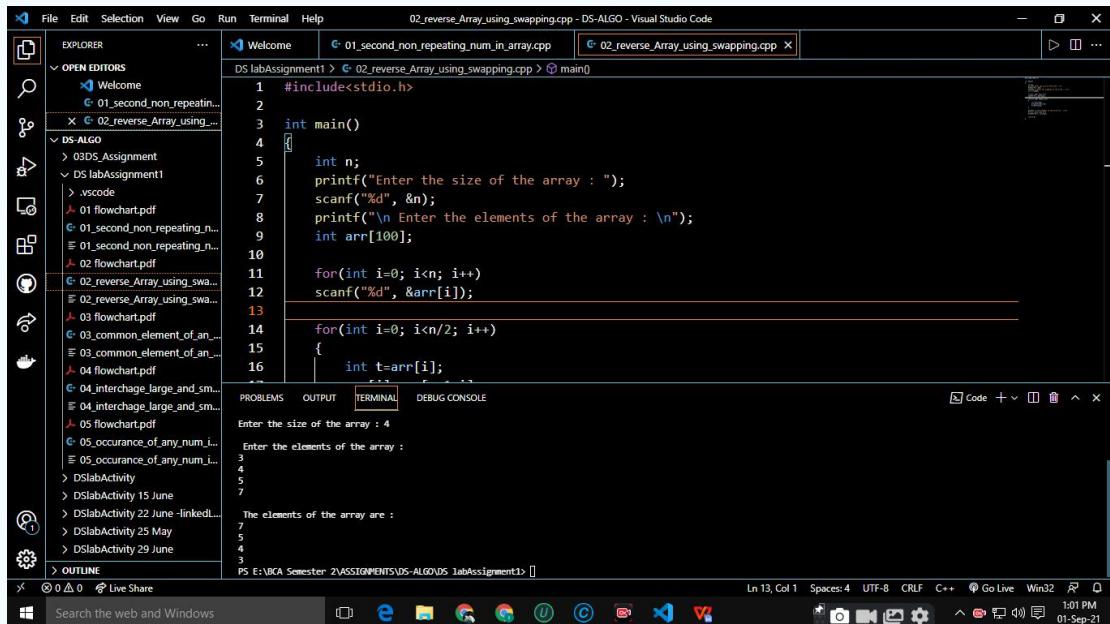
```

#include<stdio.h>

int main()
{
    int n;
    printf("Enter the size of the array : ");
    scanf("%d", &n);
    printf("\n Enter the elements of the array : \n");
    int arr[100];
    for(int i=0; i<n; i++)
        scanf("%d", &arr[i]);
    for(int i=0; i<n/2; i++)
    {
        int t=arr[i];
        arr[i]=arr[n-1-i];
        arr[n-1-i]=t;
    }
    printf("\n The elements of the array are : \n");
    for(int i=0; i<n; i++)
        printf("%d\n", arr[i]);
}

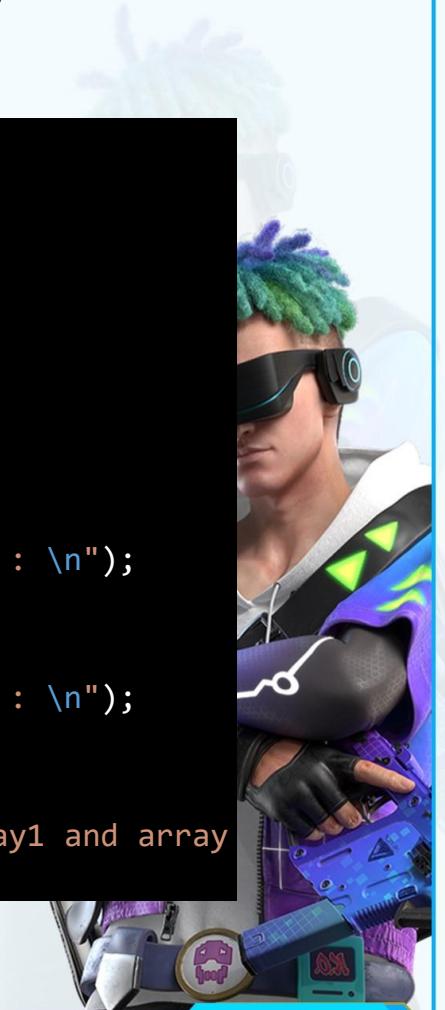
```

```
        return 0;  
}
```



```
File Edit Selection View Go Run Terminal Help 02_reverse_Array_using_swapping.cpp - DS-ALGO - Visual Studio Code  
OPEN EDITORS Welcome 01_second_non_repeating_num_in_array.cpp 02_reverse_Array_using_swapping.cpp  
DS labAssignment1 > 01_second_non_repeating_num_in_array.cpp > 02_reverse_Array_using_swapping.cpp  
1 #include<stdio.h>  
2  
3 int main()  
4 {  
5     int n;  
6     printf("Enter the size of the array : ");  
7     scanf("%d", &n);  
8     printf("\n Enter the elements of the array : \n");  
9     int arr[100];  
10  
11    for(int i=0; i<n; i++)  
12        scanf("%d", &arr[i]);  
13  
14    for(int i=0; i<n/2; i++)  
15    {  
16        int t=arr[i];  
17        arr[i]=arr[n-1-i];  
18        arr[n-1-i]=t;  
19    }  
20  
21    printf("\n The elements of the array are : ");  
22    for(int i=0; i<n; i++)  
23        printf("%d ", arr[i]);  
24  
25    return 0;  
}  
PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE  
Enter the size of the array : 4  
Enter the elements of the array :  
3  
4  
5  
7  
The elements of the array are :  
7  
6  
5  
4  
3  
Ln 13, Col 1 Spaces: 4 UTF-8 CRLF C++ Go Live Win32 10:01 PM 01-Sep-21
```

03_common_element_of_an_Array



```
#include<stdio.h>  
  
int main()  
{  
    int n;  
    int arr1[100], arr2[100];  
    printf("Enter the size of the array : ");  
    scanf("%d", &n);  
  
    printf("\n Enter the elements of the array1 : \n");  
    for(int i=0; i<n; i++)  
        scanf("%d", &arr1[i]);  
    printf("\n Enter the elements of the array2 : \n");  
    for(int i=0; i<n; i++)  
        scanf("%d", &arr2[i]);  
    printf("\n \nThe common elements of the array1 and array  
2 are : \n");
```

```
for(int i=0; i<n; i++)
{
    for(int j=0; j<n; j++)
        if(arr1[i]==arr2[j])
            printf("%d\n", arr1[i]);
}
return 0;
}
```

04_interchage_large_and_small_in_Array



```
#include<stdio.h>

int main()
{
    int n;
    int arr1[100];
    printf("Enter the size of the array : ");
    scanf("%d", &n);

    printf("\n Enter the elements of the array : \n");
    for(int i=0; i<n; i++)
        scanf("%d", &arr1[i]);
}
```

```
printf("\n\n\n The elements of the array : \n");
for(int i=0; i<n; i++)
printf("%d\n", arr1[i]);
```

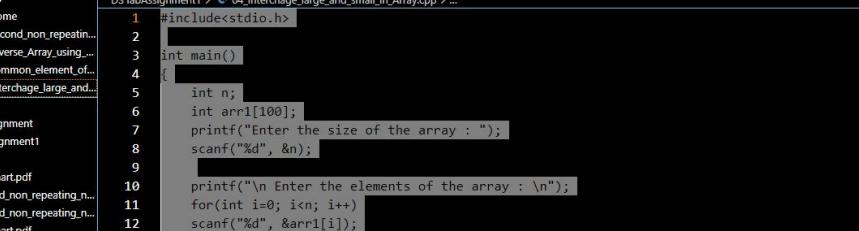
```
int *small=arr1, *large=arr1;
```

```
for(int i=0; i<n; i++)
{
    if(arr1[i]<*small)small=&arr1[i];
    if(arr1[i]>*large)large=&arr1[i];
}
```

```
int temp=*small;  
*small=*large;  
*large=temp;
```

```
printf("\n\nAfter Interchanging !!!!\n");
for(int i=0; i<n; i++)
printf("%d\n", arr1[i]);
```

```
    return 0;  
}
```



```
#include<stdio.h>
int main()
{
    int n;
    int arr1[100];
    printf("Enter the size of the array : ");
    scanf("%d", &n);
    for(int i=0; i<n; i++)
        scanf("%d", &arr1[i]);
    printf("\n\n The elements of the array : \n");
    for(int i=0; i<n; i++)
        printf("%d\n", arr1[i]);
}
```

PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE

```
The elements of the array :
21
3
5
3

After Interchanging !!!!
```

PS E:\VCA Semester 2\ASSIGNMENTS\DS_ALGO\DS labAssignment1>

05_occurrence_of_any_num_in_Array

```
#include<stdio.h>
int main()
{
    int n;
    int arr1[100];
    printf("Enter the size of the array : ");
    scanf("%d", &n);
    printf("\n Enter the elements of the array : \n");
    for(int i=0; i<n; i++)
        scanf("%d", &arr1[i]);
    printf("\n\n\n The elements of the array : \n");
    for(int i=0; i<n; i++)
        printf("%d\n", arr1[i]);
    int x, count=0;
    printf("Enter the number whose occurrence you wanna find : ");
    scanf("%d", &x);
    for(int i=0; i<n; i++)
        if(arr1[i]==x)
            count++;
    printf("The number %d occurs %d times in the array !!!!", x, count);
    return 0;
}
```





A screenshot of Visual Studio Code showing a C++ code editor. The code is for a program to find the occurrence of a number in an array. The code includes input prompts for array size and elements, and a search loop for a target number. The terminal shows the execution of the program and its output.

```
#include<stdio.h>
int main()
{
    int n;
    int arr1[100];
    printf("Enter the size of the array : ");
    scanf("%d", &n);
    printf("\nEnter the elements of the array : \n");
    for(int i=0; i<n; i++)
        scanf("%d", &arr1[i]);
    printf("\n\nThe elements of the array : \n");
    for(int i=0; i<n; i++)
        printf("%d\n", arr1[i]);
    int x, count=0;
    printf("Enter the number whose occurrence you wanna find : ");
    scanf("%d", &x);
    for(int i=0; i<n; i++)
        if(arr1[i]==x)
            count++;
    printf("The number %d occurs %d times in the array !!!\n", x, count);
}
```

PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE

The elements of the array :
3
5
7
4
Enter the number whose occurrence you wanna find : 7
The number 7 occurs 1 times in the array !!!

Ln 22, Col 2 Spaces: 4 UTF-8 CRLF C++ Go Live Win32

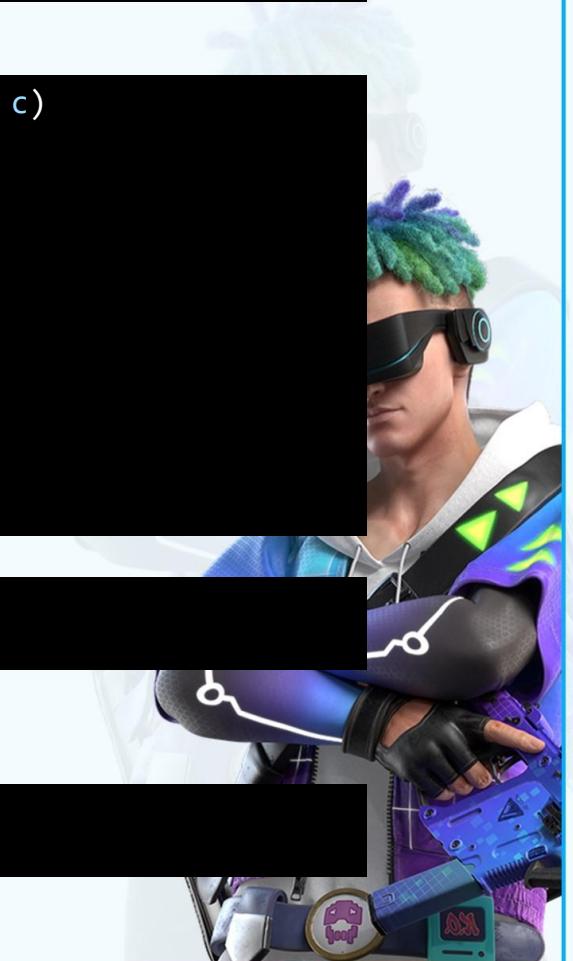
Lab Assignment 2

01_multiplicationOfMatrices

```
#include<stdio.h>
void mat_elem(int a[10][10], int r, int c)
{
    for(int i =0; i<r; i++)
    {
        for(int j=0; j<c; j++)
        {
            printf("Element [%d][%d] : ", i+1, j+1);
            scanf("%d", &a[i][j]);
        }
    }
}
```

```
void show_elem(int a[10][10], int r, int c)
{
    for(int i =0; i<r; i++)
    {
        for(int j=0; j<c; j++)
        {
            printf("%d\t ", a[i][j]);
        }
        printf("\n");
    }
}
```

```
int main()
{
```



```
int a1[10][10], r1, c1;
int a2[10][10], r2, c2;
int a3[10][10];
```

```
printf("Enter the number of elements in the row & column
of matrix 1 : \n");
scanf("%d", &r1);
scanf("%d", &c1);
```

```
printf("Enter the number of elements in the row & column
of matrix 2 : \n");
scanf("%d", &r2);
scanf("%d", &c2);
```

```
if(c1!=r2)
{
    printf("CAN'T MULTIPLY !!!!!!!\n");
    return 1;
}
```

```
printf("\n\nEnter the elements of matrix 1 : \n");
mat_elem(a1, r1, c1);
printf("\n\n\nThe matrix is : \n\n");
show_elem(a1, r1, c1);
```

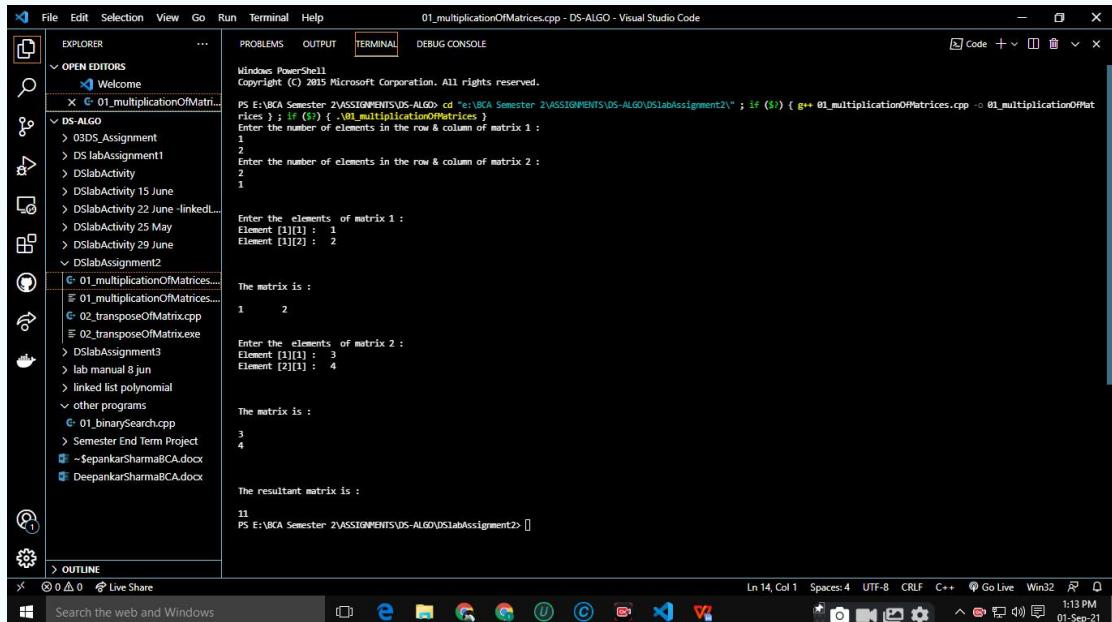
```
printf("\n\nEnter the elements of matrix 2 : \n");
mat_elem(a2, r2, c2);
printf("\n\n\nThe matrix is : \n\n");
show_elem(a2, r2, c2);
```

```
for(int i=0; i<r1; i++)
{
    for(int j=0; j<c2; j++)
    {
        a3[i][j]=0;
        for(int k=0; k<c1; k++)
            a3[i][j]+=a1[i][k]*a2[k][j];
    }
}

printf("\n\n\nThe resultant matrix is : \n\n");
```



```
    show_elem(a3,r1,c2);
}
```



```
File Edit Selection View Go Run Terminal Help 01_multiplicationOfMatrices.cpp - DS-ALGO - Visual Studio Code
PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE
Windows PowerShell
Copyright (C) 2015 Microsoft Corporation. All rights reserved.
PS E:\VNUA Semester 2\ASSIGNMENTS\DS-ALGO> cd "E:\VNUA Semester 2\ASSIGNMENTS\DS-ALGO\DSlabAssignment2\" ; g++ 01_multiplicationOfMatrices.cpp -o 01_multiplicationOfMatrices
Enter the number of elements in the row & column of matrix 1 :
1
2
Enter the number of elements in the row & column of matrix 2 :
2
1

Enter the elements of matrix 1 :
Element [1][1] : 1
Element [1][2] : 2

The matrix is :
1 2

Enter the elements of matrix 2 :
Element [1][1] : 3
Element [2][1] : 4

The matrix is :
3
4

The resultant matrix is :
11
PS E:\VNUA Semester 2\ASSIGNMENTS\DS-ALGO\DSlabAssignment2>
```

02 transposeOfMatrix

```
#include<stdio.h>
void mat_elem(int a[10][10], int r, int c)
{
    for(int i =0; i<r; i++)
    {
        for(int j=0; j<c; j++)
        {
            printf("Element [%d][%d] : ", i+1, j+1);
            scanf("%d", &a[i][j] );
        }
    }
}
```

```

void show_elem(int a[10][10], int r, int c)
{
    for(int i =0; i<r; i++)
    {
        for(int j=0; j<c; j++)
        {
            printf("%d\t ", a[i][j]);
        }
        printf("\n");
    }
}

```

```

void tps_mat(int a[10][10], int r, int c)
{
    printf("\n\nThe transpose of the matrix is : \n");
    for(int i =0; i<r; i++)
    {
        for(int j=0; j<c; j++)
        {
            printf("%d\t ", a[j][i]);
        }
        printf("\n");
    }
}

```

```

int main()
{
    int a1[10][10], r1, c1;
    // int a2[10][10], r2, c2;
    // int a3[10][10];

```

```

    printf("Enter the number of elements in the row & column
of matrix 1 : \n");
    scanf("%d",&r1);
    scanf("%d", &c1);

```

```

    // printf("Enter the number of elements in the row & col
umn of matrix 2 : \n");
    // scanf("%d",&r2);
    // scanf("%d", &c2);

    // if(c1!=r2)
    // {
    //     printf("CAN'T MULTIPLY !!!!!!!\n");
    //     return 1;
    // }

printf("\n\nEnter the elements of matrix 1 : \n");
mat_elem (a1 ,r1,c1);
printf("\n\n\nThe matrix is : \n\n");
show_elem(a1,r1,c1);
tps_mat(a1,r1,c1);

// printf("\n\nEnter the elements of matrix 2 : \n");
// mat_elem(a2,r2,c2);
// printf("\n\n\nThe matrix is : \n\n");
// show_elem(a2,r2,c2);

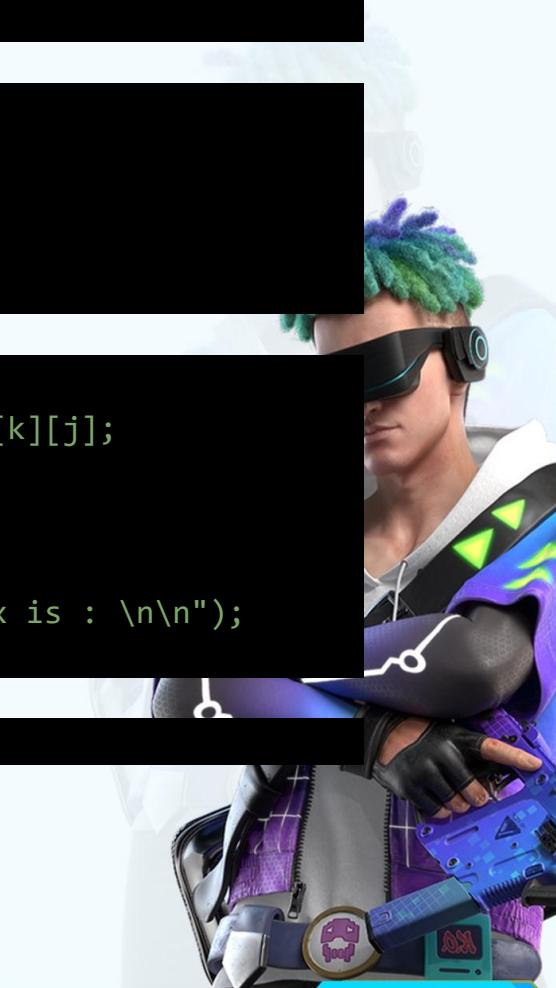
// for(int i=0; i<r1; i++)
// {
//     for(int j=0; j<c2; j++)
//     {
//         a3[i][j]=0;

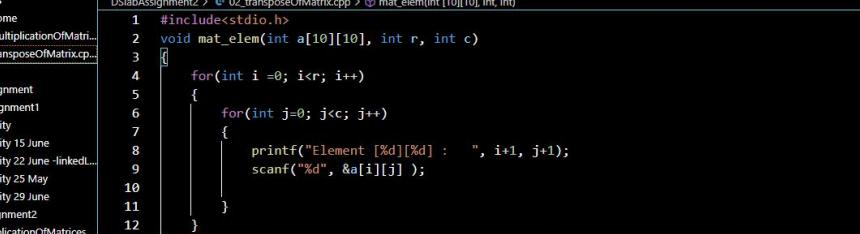
//         for(int k=0; k<c1;k++ )
//             a3[i][j]+=a1[i][k]*a2[k][j];
//     }
// }

// printf("\n\n\nThe resultant matrix is : \n\n");
// show_elem(a3,r1,c2);

}

```





The screenshot shows a Visual Studio Code interface with the following details:

- File Explorer:** Shows a tree view of files and folders, including DS-ALGO, DSLabAssignment1, DSLabAssignment2, and various C/C++ source files like 01_multiplicationOfMatrices.cpp and 02_transposeOfMatrix.cpp.
- Code Editor:** Displays the content of 02_transposeOfMatrix.cpp. The code includes a function `mat_elem` for printing matrix elements and a `show_elem` function for displaying the transpose of a matrix.
- Terminal:** Shows the output of the program, which prints the matrix and its transpose.
- Status Bar:** Shows the file path as E:\VBA Semester 2\ASSIGNMENTS\DS-ALGO\DSLabAssignment2>, line 13, column 2, and other system information.

```
#include<stdio.h>
void mat_elem(int a[10][10], int r, int c)
{
    for(int i = 0; i < r; i++)
    {
        for(int j = 0; j < c; j++)
        {
            printf("Element [%d][%d] : ", i+1, j+1);
            scanf("%d", &a[i][j]);
        }
    }
}
void show_elem(int a[10][10], int r, int c)
{
    printf("The matrix is :\n");
    for(int i = 0; i < r; i++)
    {
        for(int j = 0; j < c; j++)
        {
            printf("%d ", a[i][j]);
        }
        printf("\n");
    }
    printf("\n");
    printf("The transpose of the matrix is :\n");
    for(int i = 0; i < r; i++)
    {
        for(int j = 0; j < c; j++)
        {
            printf("%d ", a[j][i]);
        }
        printf("\n");
    }
}
```

The matrix is :

1	2
3	4

The transpose of the matrix is :

1	3
2	4

Lab Assignment 3

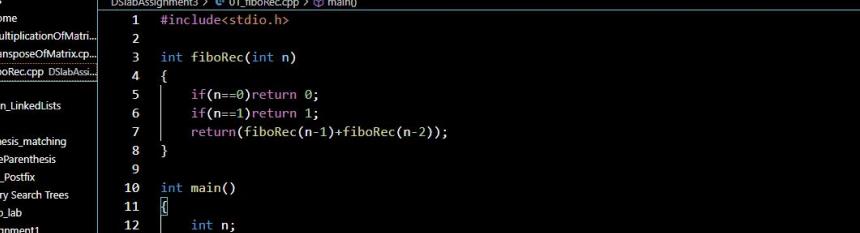
01_fibonacci_using_Recursion

```
#include<stdio.h>

int fiboRec(int n)
{
    if(n==0) return 0;
    if(n==1) return 1;
    return(fiboRec(n-1)+fiboRec(n-2));
}
```

```
int main()
{
    int n;
    printf("Enter the number : ");
    scanf("%d", &n);
    printf("The series upto %d is: \n", n);
```

```
for(int i=0; i<=n; i++)
{
    int z = fiboRec(i);
    printf("%d\t",z );
}
return 0;
}
```



```
1 #include<stdio.h>
2
3 int fiboRec(int n)
4 {
5     if(n==0) return 0;
6     if(n==1) return 1;
7     return(fiboRec(n-1)+fiboRec(n-2));
8 }
9
10 int main()
11 {
12     int n;
13     printf("Enter the number : ");
14     scanf("%d", &n);
15     printf("The series upto %d is: \n", n);
16     for(int i=0; i<=n; i++)
17     {
18         printf("%d ", fiboRec(i));
19     }
20 }
```

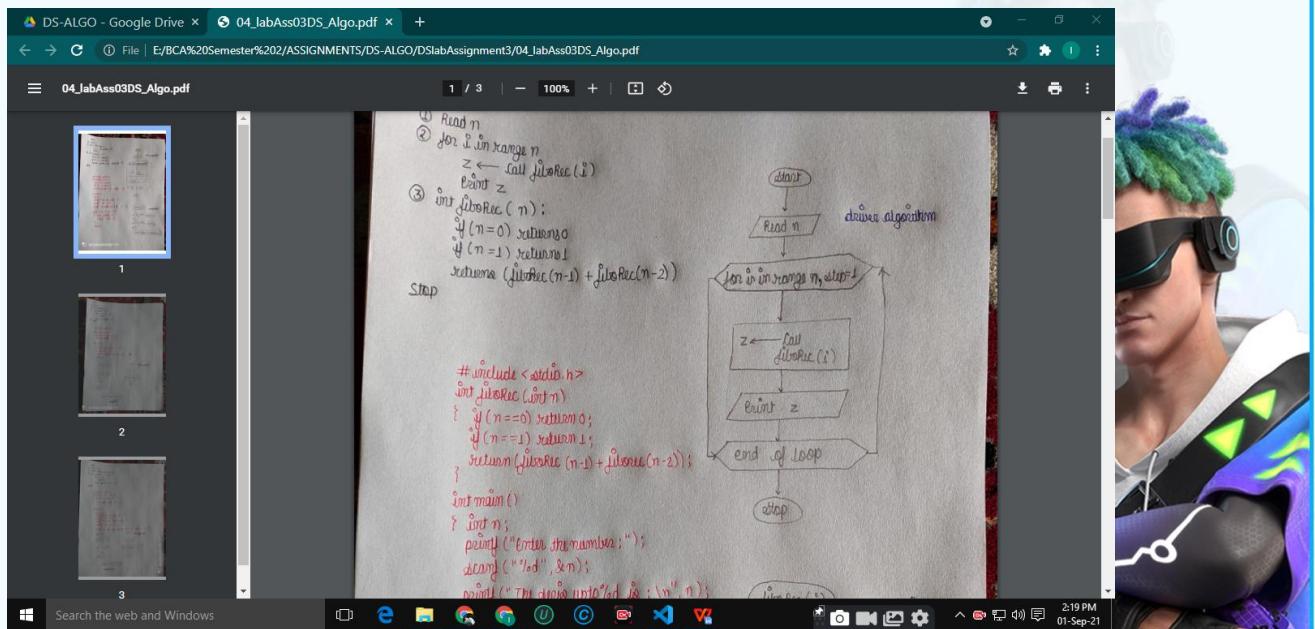
PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE

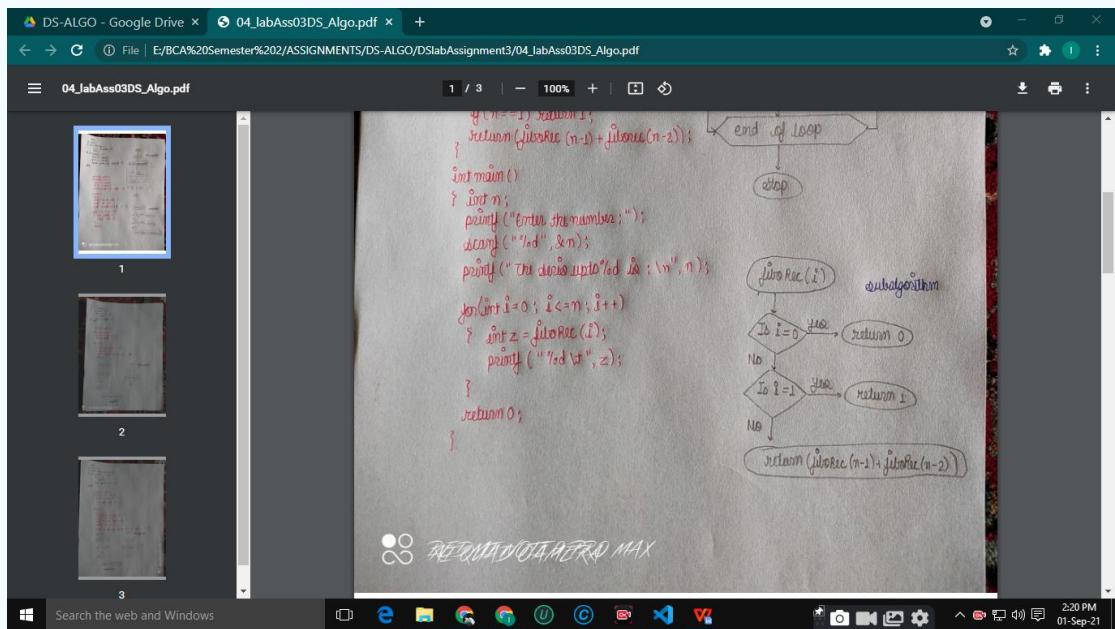
Windows PowerShell

Copyright (C) 2015 Microsoft Corporation. All rights reserved.

```
PS E:\BCA Semester 2\ASSIGNMENTS\DS-ALGO> cd "E:\BCA Semester 2\ASSIGNMENTS\DS-ALGO\DSlabAssignment3\"; if ($?) { g++ 01_fibRec.cpp -o 01_fibRec } ; if ($?) { .\01_fibRec }
```

Enter the number : 4
The series upto 4 is:
0 1 1 2 3 PS E:\BCA Semester 2\ASSIGNMENTS\DS-ALGO\DSlabAssignment3>]



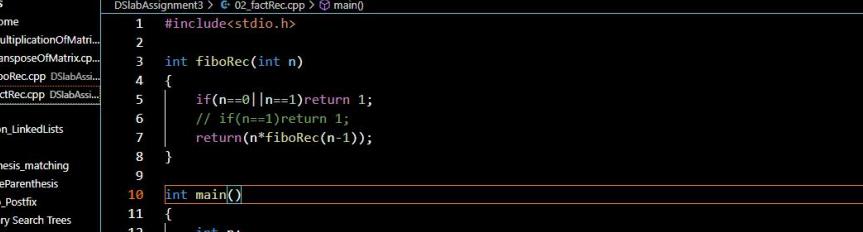


02_factRec

```
#include<stdio.h>

int fiboRec(int n)
{
    if(n==0 || n==1) return 1;
    // if(n==1) return 1;
    return(n*fiboRec(n-1));
}

int main()
{
    int n;
    printf("Enter the number : ");
    scanf("%d", &n);
    printf("The factorial of %d is: %d \n", n,fiboRec(n));
    // for(int i=0; i<=n; i++)
    // {
    //     int z = fiboRec(i);
    //     printf("%d\t",z );
    // }
    return 0;
}
```



File Edi Selection View Go Run Terminal Help 02_factRec.cpp - DS-ALGO - Visual Studio Code

EXPLORER ...

OPEN EDITORS

- Welcome
- 01_multiplicationOfMatrices.cpp
- 02_transposeOfMatrix.cpp
- 01_fiboRec.cpp
- 02_factRec.cpp DSLabAssignment3

DS-ALGO

- > 15_Insertion_LinkedLists
- > 24_stack
- > 33_parenthesis_matching
- > 34_multipleParenthesis
- > 37_Infix_To_Postfix
- > 70-80_Binary Search Trees
- > 078.dsAlgo_lab
- > DS labAssignment1
- > DS labActivity
- > DS labActivity 15 June
- > DS labActivity 22 June -linked...
- > DS labActivity 25 May
- > DS labActivity 29 June
- > DS labAssignment2
- > DS labAssignment3

DSLabAssignment3

- > vscode
- > 01_fiboRec.cpp
- > 01_fiboRec.exe
- > 02_factRec.cpp
- > 02_factRec.exe
- > 03_goldRec.cpp

OUTLINE

DSLabAssignment3 > 02_factRec.cpp > main()

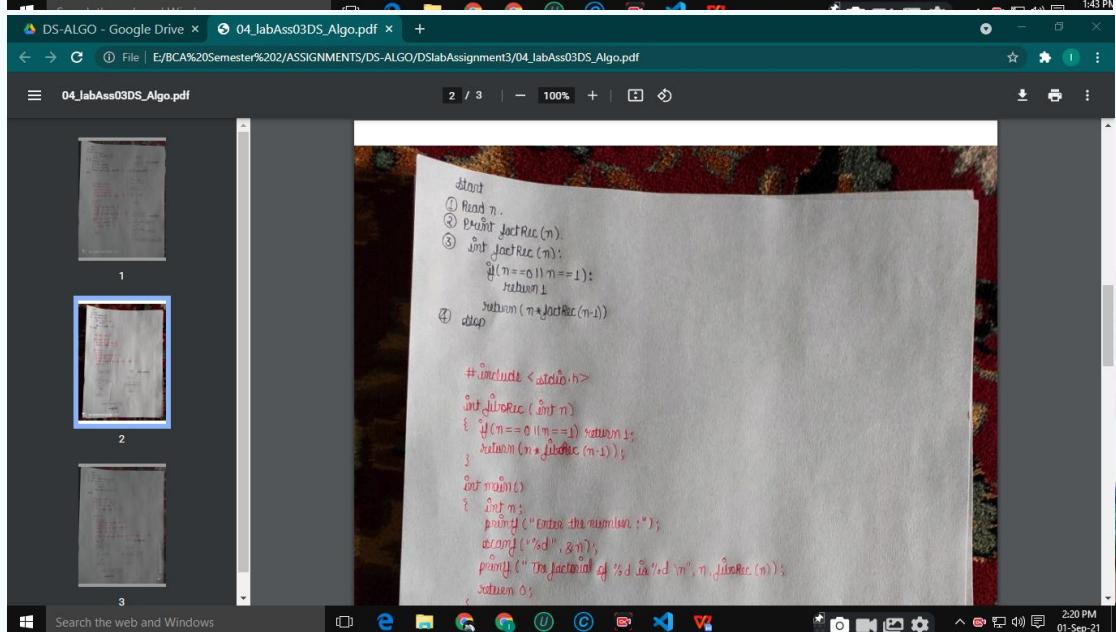
```
1 #include<stdio.h>
2
3 int fiboRec(int n)
4 {
5     if(n==0||n==1) return 1;
6     // if(n==1) return 1;
7     return(n*fiboRec(n-1));
8 }
9
10 int main()
11 {
12     int n;
13     printf("Enter the number : ");
14     scanf("%d", &n);
15     printf("The factorial of %d is: %d \n", n,fiboRec(n));
16     // for(int i=0; i<=n; i++)
17 }
```

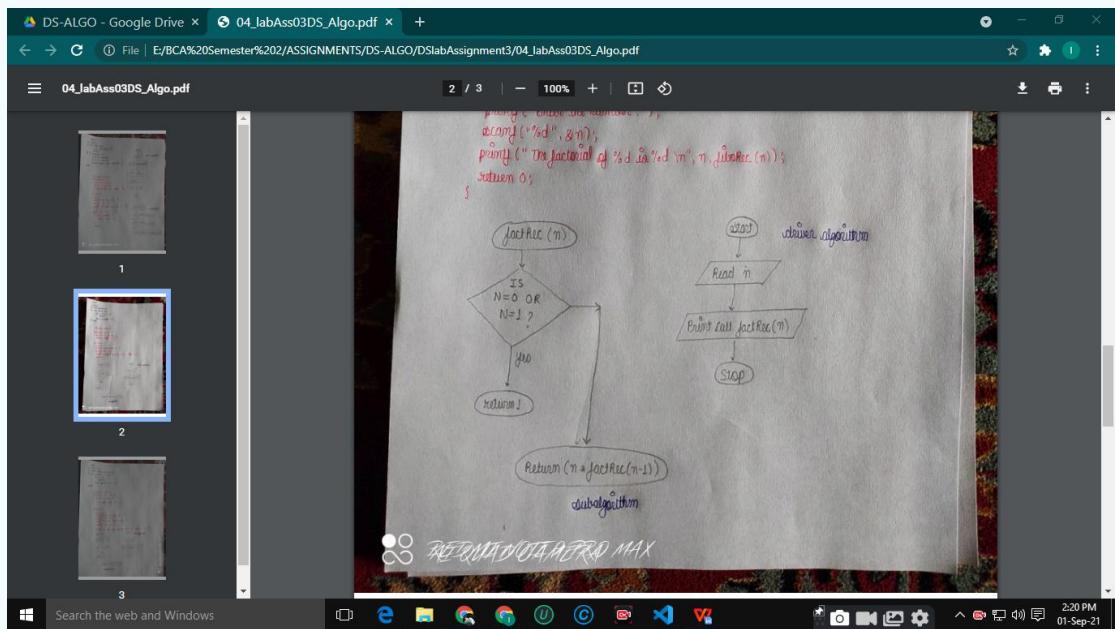
PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE

Windows PowerShell
Copyright (C) 2015 Microsoft Corporation. All rights reserved.

```
PS E:\BCA Semester 2\ASSIGNMENTS\DS-ALGO> cd "E:\BCA Semester 2\ASSIGNMENTS\DS-ALGO\DSlabAssignment3\"; if ($?) { g++ 01_fiboRec.cpp -o 01_fiboRec
Enter the number : 4
The factorial upto 4 is:
0 1 1 2 3 PS E:\BCA Semester 2\ASSIGNMENTS\DS-ALGO\DSlabAssignment3> cd "E:\BCA Semester 2\ASSIGNMENTS\DS-ALGO\DSlabAssignment3\"; if ($?) { g++ 02_factRec.cpp -o 02_factRec ; if ($?) { ./02_factRec }
Enter the number : 4
The factorial of 4 is: 24
PS E:\BCA Semester 2\ASSIGNMENTS\DS-ALGO\DSlabAssignment3>
```

Ln 10 Col 11 Spaces:4 UFT-8 CRLF C++ Go Live Win32





03_gcdRec

```
#include<stdio.h>

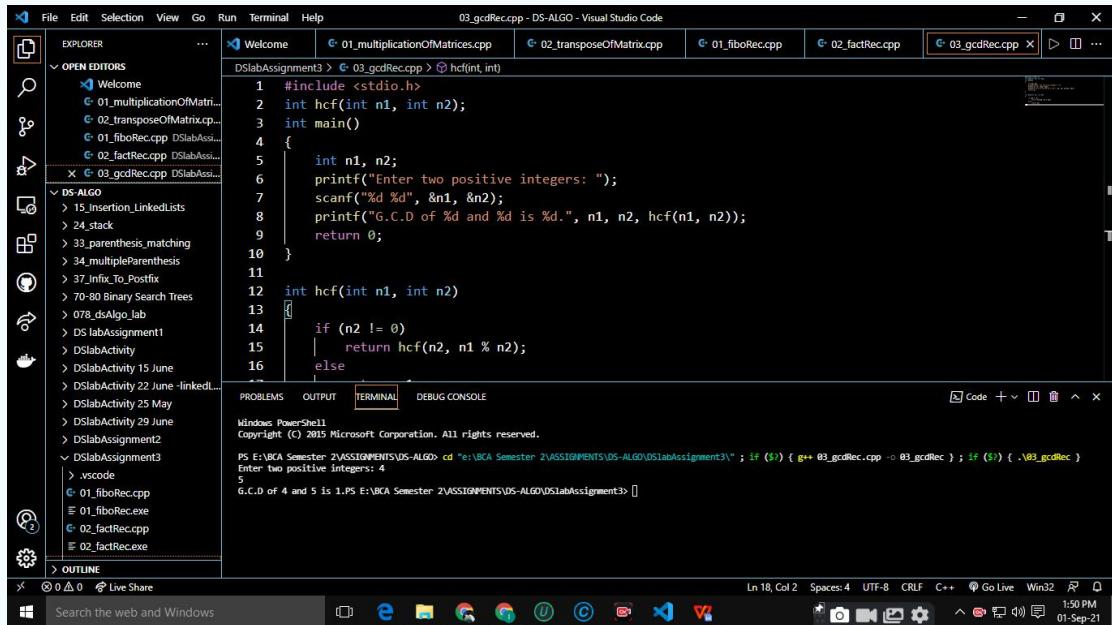
int fiboRec(int n, int m)
{
    if(n==0) return m;
    // if(n==1) return 1;
    return fiboRec(m - n, n);
}
```

```
int main()
{
    int n1,n2;
    printf("Enter the first number : ");
    scanf("%d", &n1);
    printf("Enter the second number : ");
    scanf("%d", &n2);
    printf("The Greatest Common Divisor of the two number is
%d", fiboRec(n1,n2));
    // for(int i=0; i<=n; i++)
    // {
    //     int z = fiboRec(i);
    //     printf("%d\t",z );
}
```

```

    // }
    return 0;
}

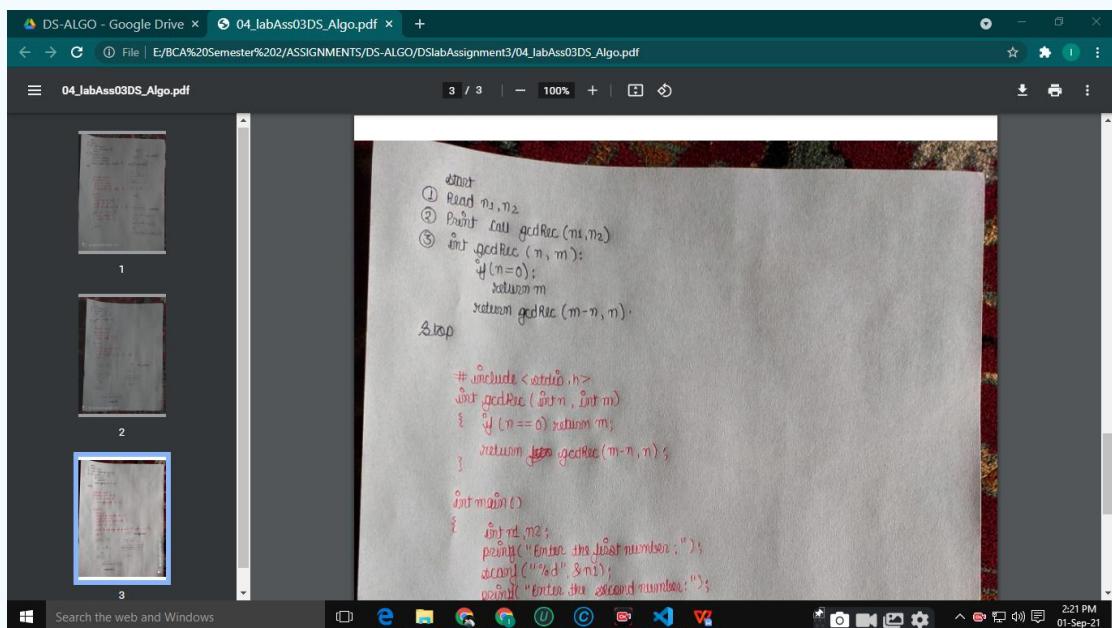
```

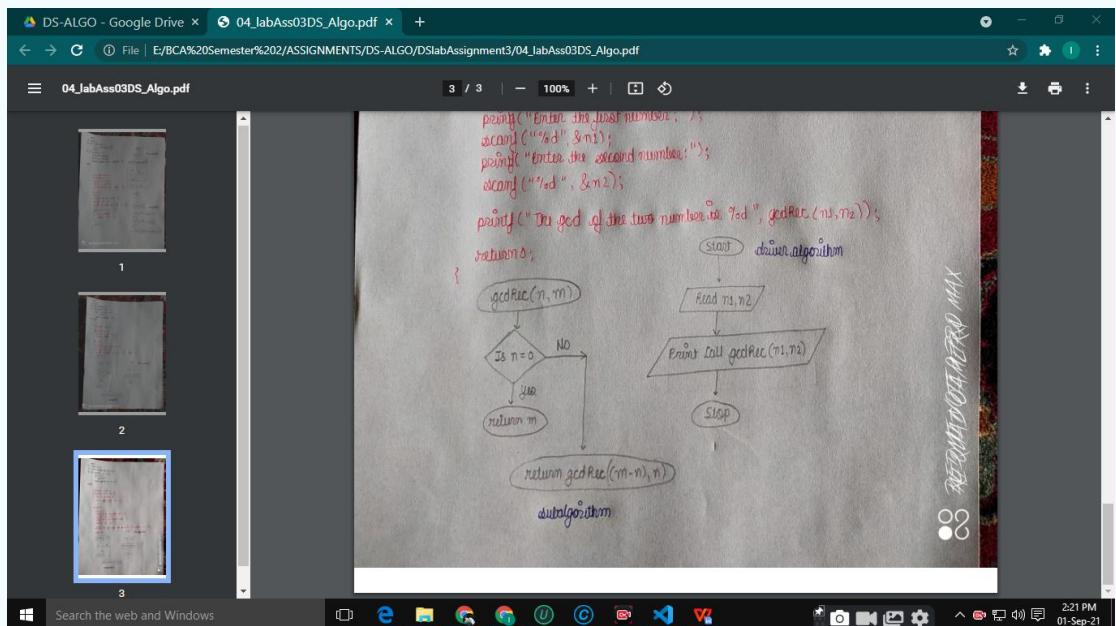


```

1 #include <stdio.h>
2 int hcf(int n1, int n2);
3 int main()
4 {
5     int n1, n2;
6     printf("Enter two positive integers: ");
7     scanf("%d %d", &n1, &n2);
8     printf("G.C.D of %d and %d is %d.", n1, n2, hcf(n1, n2));
9     return 0;
10 }
11
12 int hcf(int n1, int n2)
13 {
14     if (n2 != 0)
15     {
16         return hcf(n2, n1 % n2);
17     }
18     else
19     {
20         return n1;
21     }
22 }

```





Lab Assignment 4

01_ArrayInsertion_StaticArray

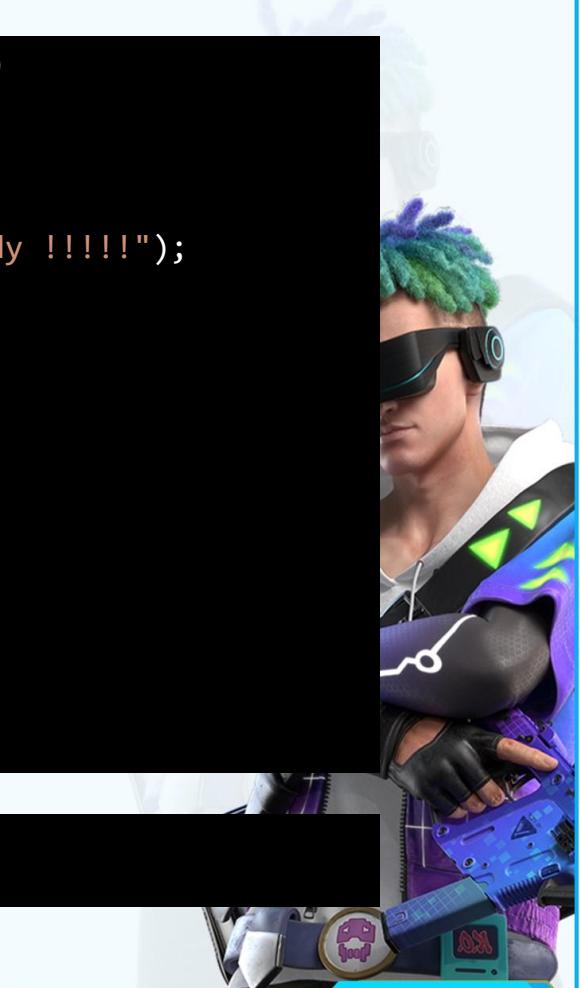
```
#include<stdio.h>

void show(int arr[], int n){

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

int ins(int *La, int n, int item, int k)
{
    if(n>=100)
    {
        printf("The array is full Already !!!!!");
        return -1;
    }
    int j =n-1;
    while(j>=k)
    {
        La[j+1] = La[j];
        j = j-1;
    }
    La[k] = item;
    return 1;
}

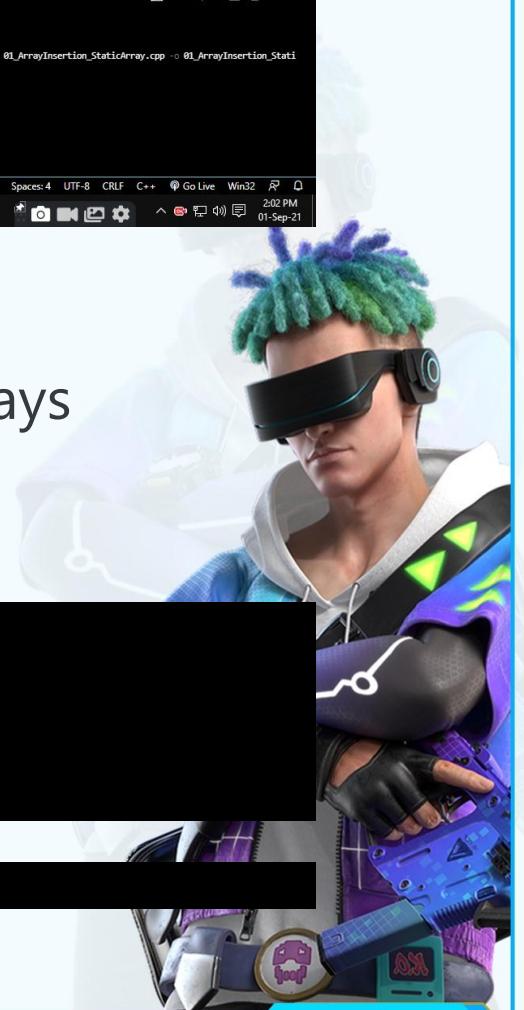
int main()
{
```



```

        int arr[100] = {7, 8, 12, 27, 88};
        int size = 5, element = 45, index=1;
        show(arr, size);
        ins(arr, size, element, index);
        size +=1;
        show(arr, size);
        return 0;
    }
}

```



A screenshot of the Visual Studio Code interface. The title bar says "01_ArrayInsertion_StaticArray.cpp - DS-ALGO - Visual Studio Code". The Explorer sidebar shows a file structure with "OPEN EDITORS" containing "01_ArrayInsertion_StaticArray.cpp" and "01_ArrayInsertion_StaticArray.h". The "DS-ALGO" folder contains various files like "01_Insertion_LinkedLists.cpp", "02_stack.cpp", etc. The "TERMINAL" tab shows a Windows PowerShell session with the following commands and output:

```

Windows PowerShell
Copyright (C) 2015 Microsoft Corporation. All rights reserved.

PS E:\VCA Semester 2\ASSIGNMENTS\DS-ALGO> cd "E:\VCA Semester 2\ASSIGNMENTS\DS-ALGO\DSlabActivity\" ; if ($_) { g++ 01_ArrayInsertion_StaticArray.cpp -o 01_ArrayInsertion_StaticArray }
01_ArrayInsertion_StaticArray.cpp
PS E:\VCA Semester 2\ASSIGNMENTS\DS-ALGO\DSlabActivity>

```

02_ArrayInsertion_DynamicArrays

```

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

void show(int *arr, int n){

    for (int i = 0; i < n; i++)

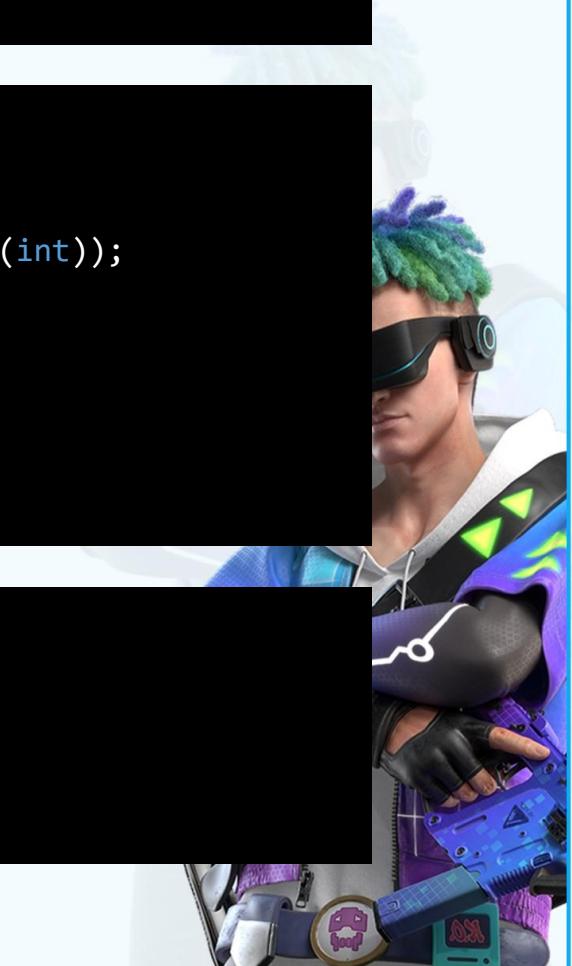
```

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

```
int ins(int *La, int n, int item, int k)
{
    if(n>=100)
    {
        printf("The array is full Already !!!!!");
        return -1;
    }
    int j =n-1;
    while(j>=k)
    {
        La[j+1] = La[j];
        j = j-1;
    }
    La[k] = item;
    return 1;
}
```

```
int main()
{
    int * arr = (int*) malloc(100*sizeof(int));
    arr[0] = 6;
    arr[1] = 56;
    arr[2] = 45;
    arr[3] = 5;
    arr[4] = 234;
    arr[5] = 55;

    int size = 5, element = 45, index=1;
    show(arr, size);
    ins(arr, size, element, index);
    size +=1;
    show(arr, size);
    return 0;
}
```



}

03_ArrayDeletion_StaticArrays

```
#include<stdio.h>

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

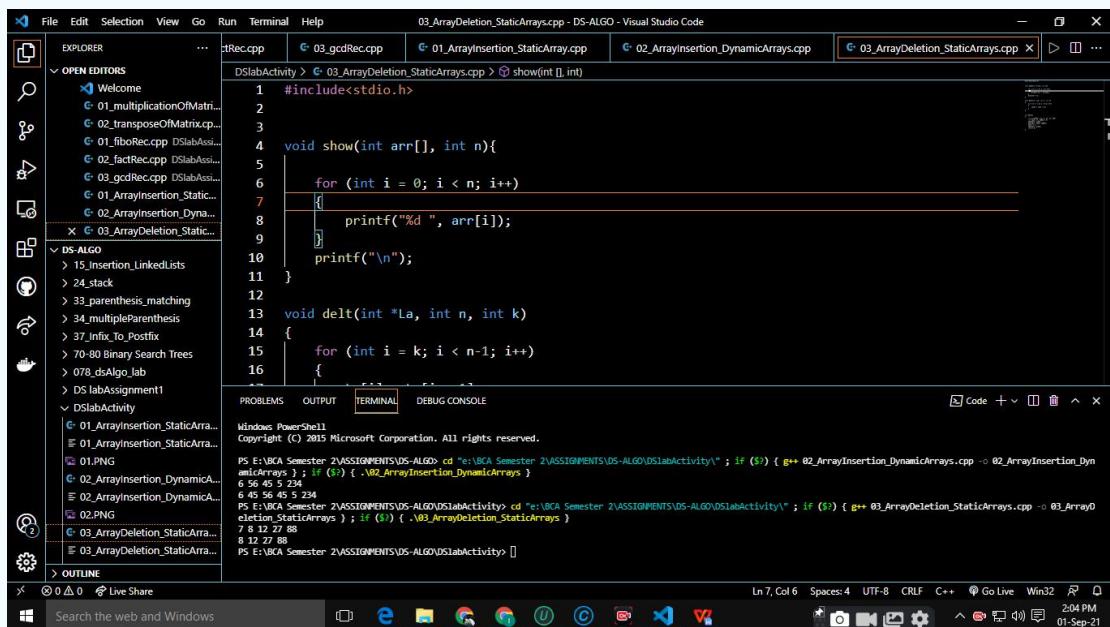
void delt(int *La, int n, int k)
{
    for (int i = k; i < n-1; i++)
    {

```



```
    La[i] = La[i + 1];  
}
```

```
int main()
{
    int arr[100] = {7, 8, 12, 27, 88};
    int size = 5, index = 0;
    show(arr, size);
    delt(arr, size, index);
    size -= 1;
    show(arr, size);
    return 0;
}
```



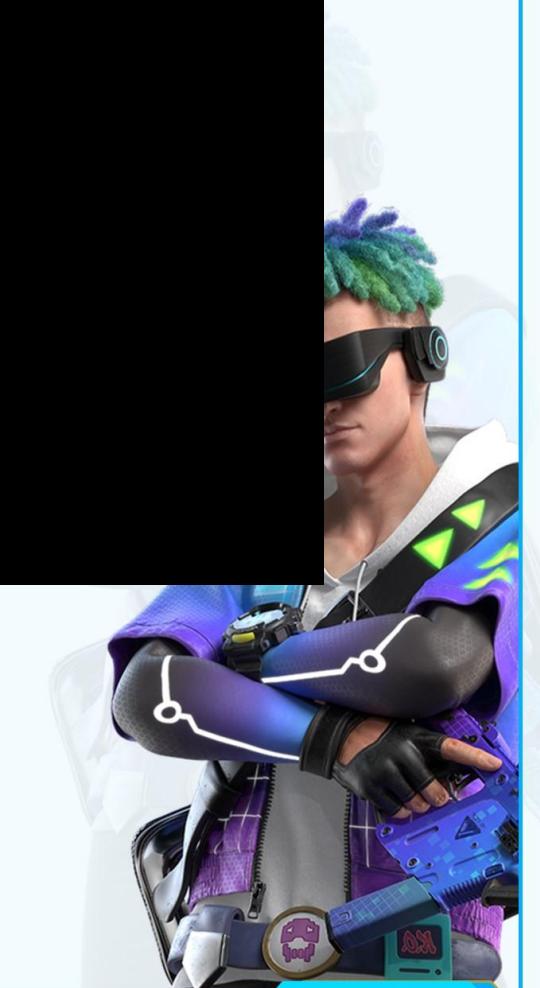
03_ArrayDeletion_DynamicArrays

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

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

```
void delt(int *La, int n, int k)  
{  
    for (int i = k; i < n-1; i++)  
    {  
        La[i] = La[i + 1];  
    }  
}
```

```
int main()  
{  
    int * arr = (int*) malloc(100*sizeof(int));  
    arr[0] = 6;  
    arr[1] = 56;  
    arr[2] = 45;  
    arr[3] = 5;  
    arr[4] = 234;  
    arr[5] = 55;  
    int size = 5, index = 5;  
    show(arr, size);  
    delt(arr, size, index);  
    size -= 1;  
    show(arr, size);  
    return 0;  
}
```



Lab Assignment 5

01_stack

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

int stack[50];
int top=-1;

// insertion
void insertion(int num)
{
    if(top==49)
    {
        printf("\nStack is full !!!!\n\n");
        return;
    }
}
```

```
        else
        {
            top=top+1;
            stack[top]=num;
        }
    }
```

```
// deletion
void deletion()
{
    if (top == -1)
    {
        printf("\nStack is already empty !!!!\n");
        return;
    }
    else
    {
        int temp=stack[top];
        top = top - 1;
        printf("\nElement deleted is %d\n", temp);
    }
}
```

```
// display
void display()
{
    if (top == -1)
    {
        printf("\nStack is empty !!!!\n");
        return;
    }
    else
    {
        for(int i=0; i<=top; i++)
        {
            printf("\n%d", stack[i]);
        }
        printf("\n");
    }
}
```

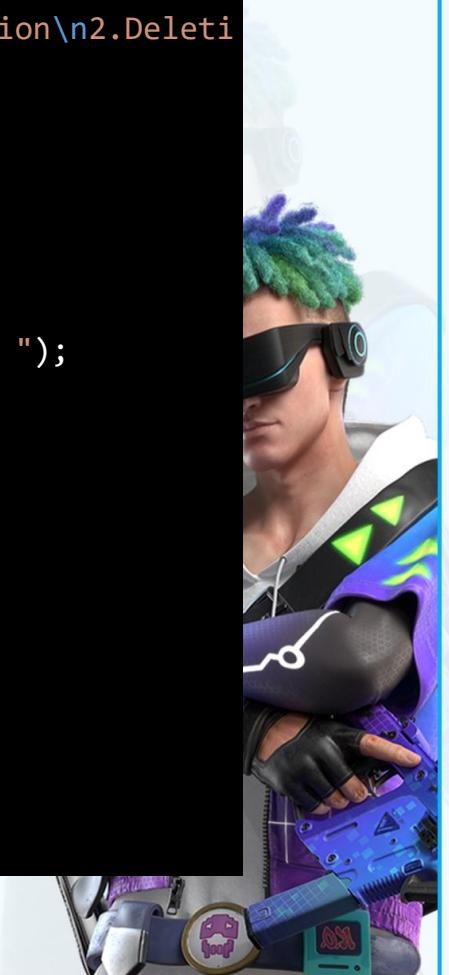


```
// peek
void peek()
{
    if (top == -1)
    {
        printf("\nStack is empty !!!\n");
        return;
    }
    else
    {
        printf("\n%d\n", stack[top]);
    }
}
```

```
int main()
{
    char ch[5]="yes";
    do
    {
        int choice;
        printf("\nSelect your choice:\n1.Insertion\n2.Deletion\n3.Display\n4.Peek\n5.exit\n");
        scanf("%d",&choice);
        switch (choice)
        {
            case 1:
                int num;
                printf("Enter the number to insert: ");
                scanf("%d", &num);
                insertion(num);
                break;

            case 2:
                deletion();
                break;

            case 3:
                display();
                break;
        }
    }
}
```



```

        case 4:
            peek();
            break;

        case 5:
            printf("\nTerminating the program !!!");
            exit(1);
            break;

    default:
        printf("Wrong choice !!!!!!!\n");

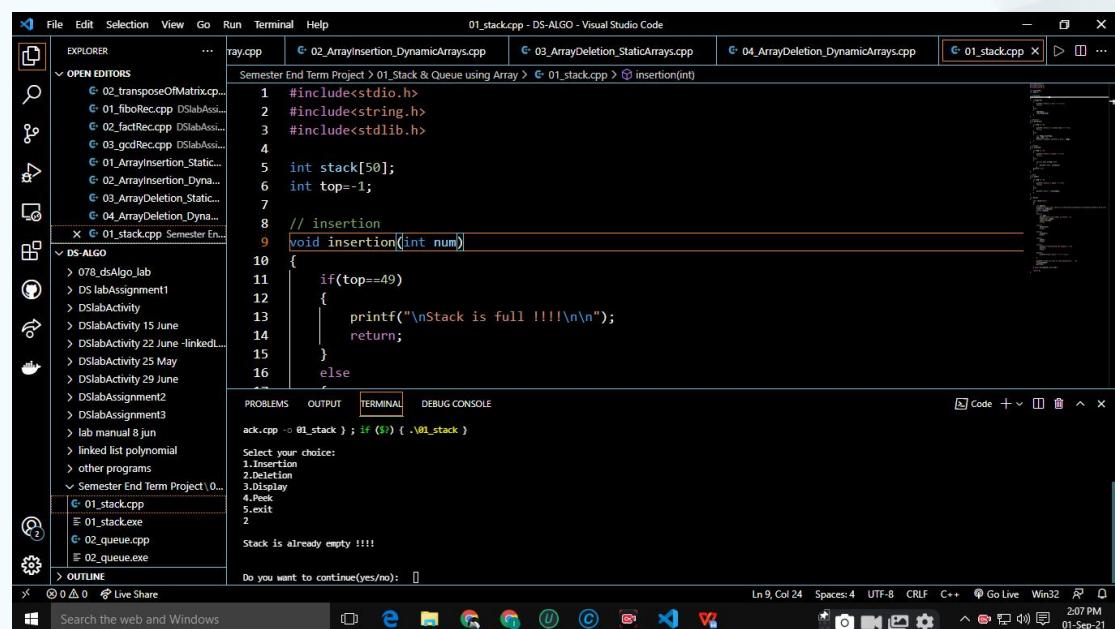
    }

printf("\n\nDo you want to continue(yes/no):   ");
fflush(stdin);
gets(ch);

} while (strcmp(ch,"yes")==0);

return 0;
}

```



```

File Edit Selection View Go Run Terminal Help
01_stack.cpp - DS-ALGO - Visual Studio Code
EXPLORER ... ray.cpp C:\02_ArrayInsertion_DynamicArrays.cpp C:\03_ArrayDeletion_StaticArrays.cpp C:\04_ArrayDeletion_DynamicArrays.cpp C:\01_stack.cpp x
OPEN EDITORS Semester End Term Project > 01_Stack & Queue using Array > 01_stack.cpp > insertion(int)
1 #include<stdio.h>
2 #include<iostream.h>
3 #include<stdlib.h>
4
5 int stack[50];
6 int top=-1;
7
8 // insertion
9 void insertion(int num)
10 {
11     if(top==49)
12     {
13         printf("\nStack is full !!!\n\n");
14         return;
15     }
16     else
17     {
18         stack[top+1]=num;
19         top=top+1;
20     }
21 }
22
23 // display
24 void display()
25 {
26     int i;
27     for(i=0;i<=top;i++)
28     {
29         printf("%d ",stack[i]);
30     }
31     printf("\n");
32 }
33
34 // peek
35 int peek()
36 {
37     if(top<0)
38     {
39         printf("Stack is empty !!!\n");
40         return -1;
41     }
42     else
43     {
44         return stack[top];
45     }
46 }
47
48 // deletion
49 void deletion()
50 {
51     if(top<0)
52     {
53         printf("Stack is empty !!!\n");
54         return;
55     }
56     else
57     {
58         top=top-1;
59     }
60 }
61
62 void insertion(int num)
63 {
64     if(top==49)
65     {
66         printf("\nStack is full !!!\n\n");
67         return;
68     }
69     else
70     {
71         stack[top+1]=num;
72         top=top+1;
73     }
74 }
75
76 // display
77 void display()
78 {
79     int i;
80     for(i=0;i<=top;i++)
81     {
82         printf("%d ",stack[i]);
83     }
84     printf("\n");
85 }
86
87 // peek
88 int peek()
89 {
90     if(top<0)
91     {
92         printf("Stack is empty !!!\n");
93         return -1;
94     }
95     else
96     {
97         return stack[top];
98     }
99 }
100
101 // deletion
102 void deletion()
103 {
104     if(top<0)
105     {
106         printf("Stack is empty !!!\n");
107         return;
108     }
109     else
110     {
111         top=top-1;
112     }
113 }
114
115 void insertion(int num)
116 {
117     if(top==49)
118     {
119         printf("\nStack is full !!!\n\n");
120         return;
121     }
122     else
123     {
124         stack[top+1]=num;
125         top=top+1;
126     }
127 }
128
129 // display
130 void display()
131 {
132     int i;
133     for(i=0;i<=top;i++)
134     {
135         printf("%d ",stack[i]);
136     }
137     printf("\n");
138 }
139
140 // peek
141 int peek()
142 {
143     if(top<0)
144     {
145         printf("Stack is empty !!!\n");
146         return -1;
147     }
148     else
149     {
150         return stack[top];
151     }
152 }
153
154 // deletion
155 void deletion()
156 {
157     if(top<0)
158     {
159         printf("Stack is empty !!!\n");
160         return;
161     }
162     else
163     {
164         top=top-1;
165     }
166 }
167
168 void insertion(int num)
169 {
170     if(top==49)
171     {
172         printf("\nStack is full !!!\n\n");
173         return;
174     }
175     else
176     {
177         stack[top+1]=num;
178         top=top+1;
179     }
180 }
181
182 // display
183 void display()
184 {
185     int i;
186     for(i=0;i<=top;i++)
187     {
188         printf("%d ",stack[i]);
189     }
190     printf("\n");
191 }
192
193 // peek
194 int peek()
195 {
196     if(top<0)
197     {
198         printf("Stack is empty !!!\n");
199         return -1;
200     }
201     else
202     {
203         return stack[top];
204     }
205 }
206
207 // deletion
208 void deletion()
209 {
210     if(top<0)
211     {
212         printf("Stack is empty !!!\n");
213         return;
214     }
215     else
216     {
217         top=top-1;
218     }
219 }
220
221 void insertion(int num)
222 {
223     if(top==49)
224     {
225         printf("\nStack is full !!!\n\n");
226         return;
227     }
228     else
229     {
230         stack[top+1]=num;
231         top=top+1;
232     }
233 }
234
235 // display
236 void display()
237 {
238     int i;
239     for(i=0;i<=top;i++)
240     {
241         printf("%d ",stack[i]);
242     }
243     printf("\n");
244 }
245
246 // peek
247 int peek()
248 {
249     if(top<0)
250     {
251         printf("Stack is empty !!!\n");
252         return -1;
253     }
254     else
255     {
256         return stack[top];
257     }
258 }
259
260 // deletion
261 void deletion()
262 {
263     if(top<0)
264     {
265         printf("Stack is empty !!!\n");
266         return;
267     }
268     else
269     {
270         top=top-1;
271     }
272 }
273
274 void insertion(int num)
275 {
276     if(top==49)
277     {
278         printf("\nStack is full !!!\n\n");
279         return;
280     }
281     else
282     {
283         stack[top+1]=num;
284         top=top+1;
285     }
286 }
287
288 // display
289 void display()
290 {
291     int i;
292     for(i=0;i<=top;i++)
293     {
294         printf("%d ",stack[i]);
295     }
296     printf("\n");
297 }
298
299 // peek
300 int peek()
301 {
302     if(top<0)
303     {
304         printf("Stack is empty !!!\n");
305         return -1;
306     }
307     else
308     {
309         return stack[top];
310     }
311 }
312
313 // deletion
314 void deletion()
315 {
316     if(top<0)
317     {
318         printf("Stack is empty !!!\n");
319         return;
320     }
321     else
322     {
323         top=top-1;
324     }
325 }
326
327 void insertion(int num)
328 {
329     if(top==49)
330     {
331         printf("\nStack is full !!!\n\n");
332         return;
333     }
334     else
335     {
336         stack[top+1]=num;
337         top=top+1;
338     }
339 }
340
341 // display
342 void display()
343 {
344     int i;
345     for(i=0;i<=top;i++)
346     {
347         printf("%d ",stack[i]);
348     }
349     printf("\n");
350 }
351
352 // peek
353 int peek()
354 {
355     if(top<0)
356     {
357         printf("Stack is empty !!!\n");
358         return -1;
359     }
360     else
361     {
362         return stack[top];
363     }
364 }
365
366 // deletion
367 void deletion()
368 {
369     if(top<0)
370     {
371         printf("Stack is empty !!!\n");
372         return;
373     }
374     else
375     {
376         top=top-1;
377     }
378 }
379
380 void insertion(int num)
381 {
382     if(top==49)
383     {
384         printf("\nStack is full !!!\n\n");
385         return;
386     }
387     else
388     {
389         stack[top+1]=num;
390         top=top+1;
391     }
392 }
393
394 // display
395 void display()
396 {
397     int i;
398     for(i=0;i<=top;i++)
399     {
400         printf("%d ",stack[i]);
401     }
402     printf("\n");
403 }
404
405 // peek
406 int peek()
407 {
408     if(top<0)
409     {
410         printf("Stack is empty !!!\n");
411         return -1;
412     }
413     else
414     {
415         return stack[top];
416     }
417 }
418
419 // deletion
420 void deletion()
421 {
422     if(top<0)
423     {
424         printf("Stack is empty !!!\n");
425         return;
426     }
427     else
428     {
429         top=top-1;
430     }
431 }
432
433 void insertion(int num)
434 {
435     if(top==49)
436     {
437         printf("\nStack is full !!!\n\n");
438         return;
439     }
440     else
441     {
442         stack[top+1]=num;
443         top=top+1;
444     }
445 }
446
447 // display
448 void display()
449 {
450     int i;
451     for(i=0;i<=top;i++)
452     {
453         printf("%d ",stack[i]);
454     }
455     printf("\n");
456 }
457
458 // peek
459 int peek()
460 {
461     if(top<0)
462     {
463         printf("Stack is empty !!!\n");
464         return -1;
465     }
466     else
467     {
468         return stack[top];
469     }
470 }
471
472 // deletion
473 void deletion()
474 {
475     if(top<0)
476     {
477         printf("Stack is empty !!!\n");
478         return;
479     }
480     else
481     {
482         top=top-1;
483     }
484 }
485
486 void insertion(int num)
487 {
488     if(top==49)
489     {
490         printf("\nStack is full !!!\n\n");
491         return;
492     }
493     else
494     {
495         stack[top+1]=num;
496         top=top+1;
497     }
498 }
499
500 // display
501 void display()
502 {
503     int i;
504     for(i=0;i<=top;i++)
505     {
506         printf("%d ",stack[i]);
507     }
508     printf("\n");
509 }
510
511 // peek
512 int peek()
513 {
514     if(top<0)
515     {
516         printf("Stack is empty !!!\n");
517         return -1;
518     }
519     else
520     {
521         return stack[top];
522     }
523 }
524
525 // deletion
526 void deletion()
527 {
528     if(top<0)
529     {
530         printf("Stack is empty !!!\n");
531         return;
532     }
533     else
534     {
535         top=top-1;
536     }
537 }
538
539 void insertion(int num)
540 {
541     if(top==49)
542     {
543         printf("\nStack is full !!!\n\n");
544         return;
545     }
546     else
547     {
548         stack[top+1]=num;
549         top=top+1;
550     }
551 }
552
553 // display
554 void display()
555 {
556     int i;
557     for(i=0;i<=top;i++)
558     {
559         printf("%d ",stack[i]);
560     }
561     printf("\n");
562 }
563
564 // peek
565 int peek()
566 {
567     if(top<0)
568     {
569         printf("Stack is empty !!!\n");
570         return -1;
571     }
572     else
573     {
574         return stack[top];
575     }
576 }
577
578 // deletion
579 void deletion()
580 {
581     if(top<0)
582     {
583         printf("Stack is empty !!!\n");
584         return;
585     }
586     else
587     {
588         top=top-1;
589     }
590 }
591
592 void insertion(int num)
593 {
594     if(top==49)
595     {
596         printf("\nStack is full !!!\n\n");
597         return;
598     }
599     else
600     {
601         stack[top+1]=num;
602         top=top+1;
603     }
604 }
605
606 // display
607 void display()
608 {
609     int i;
610     for(i=0;i<=top;i++)
611     {
612         printf("%d ",stack[i]);
613     }
614     printf("\n");
615 }
616
617 // peek
618 int peek()
619 {
620     if(top<0)
621     {
622         printf("Stack is empty !!!\n");
623         return -1;
624     }
625     else
626     {
627         return stack[top];
628     }
629 }
630
631 // deletion
632 void deletion()
633 {
634     if(top<0)
635     {
636         printf("Stack is empty !!!\n");
637         return;
638     }
639     else
640     {
641         top=top-1;
642     }
643 }
644
645 void insertion(int num)
646 {
647     if(top==49)
648     {
649         printf("\nStack is full !!!\n\n");
650         return;
651     }
652     else
653     {
654         stack[top+1]=num;
655         top=top+1;
656     }
657 }
658
659 // display
660 void display()
661 {
662     int i;
663     for(i=0;i<=top;i++)
664     {
665         printf("%d ",stack[i]);
666     }
667     printf("\n");
668 }
669
670 // peek
671 int peek()
672 {
673     if(top<0)
674     {
675         printf("Stack is empty !!!\n");
676         return -1;
677     }
678     else
679     {
680         return stack[top];
681     }
682 }
683
684 // deletion
685 void deletion()
686 {
687     if(top<0)
688     {
689         printf("Stack is empty !!!\n");
690         return;
691     }
692     else
693     {
694         top=top-1;
695     }
696 }
697
698 void insertion(int num)
699 {
700     if(top==49)
701     {
702         printf("\nStack is full !!!\n\n");
703         return;
704     }
705     else
706     {
707         stack[top+1]=num;
708         top=top+1;
709     }
710 }
711
712 // display
713 void display()
714 {
715     int i;
716     for(i=0;i<=top;i++)
717     {
718         printf("%d ",stack[i]);
719     }
720     printf("\n");
721 }
722
723 // peek
724 int peek()
725 {
726     if(top<0)
727     {
728         printf("Stack is empty !!!\n");
729         return -1;
730     }
731     else
732     {
733         return stack[top];
734     }
735 }
736
737 // deletion
738 void deletion()
739 {
740     if(top<0)
741     {
742         printf("Stack is empty !!!\n");
743         return;
744     }
745     else
746     {
747         top=top-1;
748     }
749 }
750
751 void insertion(int num)
752 {
753     if(top==49)
754     {
755         printf("\nStack is full !!!\n\n");
756         return;
757     }
758     else
759     {
760         stack[top+1]=num;
761         top=top+1;
762     }
763 }
764
765 // display
766 void display()
767 {
768     int i;
769     for(i=0;i<=top;i++)
770     {
771         printf("%d ",stack[i]);
772     }
773     printf("\n");
774 }
775
776 // peek
777 int peek()
778 {
779     if(top<0)
780     {
781         printf("Stack is empty !!!\n");
782         return -1;
783     }
784     else
785     {
786         return stack[top];
787     }
788 }
789
790 // deletion
791 void deletion()
792 {
793     if(top<0)
794     {
795         printf("Stack is empty !!!\n");
796         return;
797     }
798     else
799     {
800         top=top-1;
801     }
802 }
803
804 void insertion(int num)
805 {
806     if(top==49)
807     {
808         printf("\nStack is full !!!\n\n");
809         return;
810     }
811     else
812     {
813         stack[top+1]=num;
814         top=top+1;
815     }
816 }
817
818 // display
819 void display()
820 {
821     int i;
822     for(i=0;i<=top;i++)
823     {
824         printf("%d ",stack[i]);
825     }
826     printf("\n");
827 }
828
829 // peek
830 int peek()
831 {
832     if(top<0)
833     {
834         printf("Stack is empty !!!\n");
835         return -1;
836     }
837     else
838     {
839         return stack[top];
840     }
841 }
842
843 // deletion
844 void deletion()
845 {
846     if(top<0)
847     {
848         printf("Stack is empty !!!\n");
849         return;
850     }
851     else
852     {
853         top=top-1;
854     }
855 }
856
857 void insertion(int num)
858 {
859     if(top==49)
860     {
861         printf("\nStack is full !!!\n\n");
862         return;
863     }
864     else
865     {
866         stack[top+1]=num;
867         top=top+1;
868     }
869 }
870
871 // display
872 void display()
873 {
874     int i;
875     for(i=0;i<=top;i++)
876     {
877         printf("%d ",stack[i]);
878     }
879     printf("\n");
880 }
881
882 // peek
883 int peek()
884 {
885     if(top<0)
886     {
887         printf("Stack is empty !!!\n");
888         return -1;
889     }
890     else
891     {
892         return stack[top];
893     }
894 }
895
896 // deletion
897 void deletion()
898 {
899     if(top<0)
900     {
901         printf("Stack is empty !!!\n");
902         return;
903     }
904     else
905     {
906         top=top-1;
907     }
908 }
909
910 void insertion(int num)
911 {
912     if(top==49)
913     {
914         printf("\nStack is full !!!\n\n");
915         return;
916     }
917     else
918     {
919         stack[top+1]=num;
920         top=top+1;
921     }
922 }
923
924 // display
925 void display()
926 {
927     int i;
928     for(i=0;i<=top;i++)
929     {
930         printf("%d ",stack[i]);
931     }
932     printf("\n");
933 }
934
935 // peek
936 int peek()
937 {
938     if(top<0)
939     {
940         printf("Stack is empty !!!\n");
941         return -1;
942     }
943     else
944     {
945         return stack[top];
946     }
947 }
948
949 // deletion
950 void deletion()
951 {
952     if(top<0)
953     {
954         printf("Stack is empty !!!\n");
955         return;
956     }
957     else
958     {
959         top=top-1;
960     }
961 }
962
963 void insertion(int num)
964 {
965     if(top==49)
966     {
967         printf("\nStack is full !!!\n\n");
968         return;
969     }
970     else
971     {
972         stack[top+1]=num;
973         top=top+1;
974     }
975 }
976
977 // display
978 void display()
979 {
980     int i;
981     for(i=0;i<=top;i++)
982     {
983         printf("%d ",stack[i]);
984     }
985     printf("\n");
986 }
987
988 // peek
989 int peek()
990 {
991     if(top<0)
992     {
993         printf("Stack is empty !!!\n");
994         return -1;
995     }
996     else
997     {
998         return stack[top];
999     }
999 }
999
999 // deletion
999 void deletion()
999 {
999     if(top<0)
999     {
999         printf("Stack is empty !!!\n");
999         return;
999     }
999     else
999     {
999         top=top-1;
999     }
999 }
999
999 void insertion(int num)
999 {
999     if(top==49)
999     {
999         printf("\nStack is full !!!\n\n");
999         return;
999     }
999     else
999     {
999         stack[top+1]=num;
999         top=top+1;
999     }
999 }
999
999 // display
999 void display()
999 {
999     int i;
999     for(i=0;i<=top;i++)
999     {
999         printf("%d ",stack[i]);
999     }
999     printf("\n");
999 }
999
999 // peek
999 int peek()
999 {
999     if(top<0)
999     {
999         printf("Stack is empty !!!\n");
999         return -1;
999     }
999     else
999     {
999         return stack[top];
999     }
999 }
999
999 // deletion
999 void deletion()
999 {
999     if(top<0)
999     {
999         printf("Stack is empty !!!\n");
999         return;
999     }
999     else
999     {
999         top=top-1;
999     }
999 }
999
999 void insertion(int num)
999 {
999     if(top==49)
999     {
999         printf("\nStack is full !!!\n\n");
999         return;
999     }
999     else
999     {
999         stack[top+1]=num;
999         top=top+1;
999     }
999 }
999
999 // display
999 void display()
999 {
999     int i;
999     for(i=0;i<=top;i++)
999     {
999         printf("%d ",stack[i]);
999     }
999     printf("\n");
999 }
999
999 // peek
999 int peek()
999 {
999     if(top<0)
999     {
999         printf("Stack is empty !!!\n");
999         return -1;
999     }
999     else
999     {
999         return stack[top];
999     }
999 }
999
999 // deletion
999 void deletion()
999 {
999     if(top<0)
999     {
999         printf("Stack is empty !!!\n");
999         return;
999     }
999     else
999     {
999         top=top-1;
999     }
999 }
999
999 void insertion(int num)
999 {
999     if(top==49)
999     {
999         printf("\nStack is full !!!\n\n");
999         return;
999     }
999     else
999     {
999         stack[top+1]=num;
999         top=top+1;
999     }
999 }
999
999 // display
999 void display()
999 {
999     int i;
999     for(i=0;i<=top;i++)
999     {
999         printf("%d ",stack[i]);
999     }
999     printf("\n");
999 }
999
999 // peek
999 int peek()
999 {
999     if(top<0)
999     {
999         printf("Stack is empty !!!\n");
999         return -1;
999     }
999     else
999     {
999         return stack[top];
999     }
999 }
999
999 // deletion
999 void deletion()
999 {
999     if(top<0)
999     {
999         printf("Stack is empty !!!\n");
999         return;
999     }
999     else
999     {
999         top=top-1;
999     }
999 }
999
999 void insertion(int num)
999 {
999     if(top==49)
999     {
999         printf("\nStack is full !!!\n\n");
999         return;
999     }
999     else
999     {
999         stack[top+1]=num;
999         top=top+1;
999     }
999 }
999
999 // display
999 void display()
999 {
999     int i;
999     for(i=0;i<=top;i++)
999     {
999         printf("%d ",stack[i]);
999     }
999     printf("\n");
999 }
999
999 // peek
999 int peek()
999 {
999     if(top<0)
999     {
999         printf("Stack is empty !!!\n");
999         return -1;
999     }
999     else
999     {
999         return stack[top];
999     }
999 }
999
999 // deletion
999 void deletion()
999 {
999     if(top<0)
999     {
999         printf("Stack is empty !!!\n");
999         return;
999     }
999     else
999     {
999         top=top-1;
999     }
999 }
999
999 void insertion(int num)
999 {
999     if(top==49)
999     {
999         printf("\nStack is full !!!\n\n");
999         return;
999     }
999     else
999     {
999         stack[top+1]=num;
999         top=top+1;
999     }
999 }
999
999 // display
999 void display()
999 {
999     int i;
999     for(i=0;i<=top;i++)
999     {
999         printf("%d ",stack[i]);
999     }
999     printf("\n");
999 }
999
999 // peek
999 int peek()
999 {
999     if(top<0)
999     {
999         printf("Stack is empty !!!\n");
999         return -1;
999     }
999     else
999     {
999         return stack[top];
999     }
999 }
999
999 // deletion
999 void deletion()
999 {
999     if(top<0)
999     {
999         printf("Stack is empty !!!\n");
999         return;
999     }
999     else
999     {
999         top=top-1;
999     }
999 }
999
999 void insertion(int num)
999 {
999     if(top==49)
999     {
999         printf("\nStack is full !!!\n\n");
999         return;
999     }
999     else
999     {
999         stack[top+1]=num;
999         top=top+1;
999     }
999 }
999
999 // display
999 void display()
999 {
999     int i;
999     for(i=0;i<=top;i++)
999     {
999         printf("%d ",stack[i]);
999     }
999     printf("\n");
999 }
999
999 // peek
999 int peek()
999 {
999     if(top<0)
999     {
999         printf("Stack is empty !!!\n");
999         return -1;
999     }
999     else
999     {
999         return stack[top];
999     }
999 }
999
999 // deletion
999 void deletion()
999 {
999     if(top<0)
999     {
999         printf("Stack is empty !!!\n");
999         return;
999     }
999     else
999     {
999         top=top-1;
999     }
999 }
999
999 void insertion(int num)
999 {
999     if(top==49)
999     {
999         printf("\nStack is full !!!\n\n");
999         return;
999     }
999     else
999     {
999         stack[top+1]=num;
999         top=top+1;
999     }
999 }
999
999 // display
999 void display()
999 {
999     int i;
999     for(i=0;i<=top;i++)
999     {
999         printf("%d ",stack[i]);
999     }
999     printf("\n");
999 }
999
999 // peek
999 int peek()
999 {
999     if(top<0)
999     {
999         printf("Stack is empty !!!\n");
999         return -1;
999     }
999     else
999     {
999         return stack[top];
999     }
999 }
999
999 // deletion
999 void deletion()
999 {
999     if(top<0)
999     {
999         printf("Stack is empty !!!\n");
999         return;
999     }
999     else
999     {
999         top=top-1;
999     }
999 }
999
999 void insertion(int num)
999 {
999     if(top==49)
999     {
999         printf("\nStack is full !!!\n\n");
999         return;
999     }
999     else
999     {
999         stack[top+1]=num;
999         top=top+1;
999     }
999 }
999
999 // display
999 void display()
999 {
999     int i;
999     for(i=0;i<=top;i++)
999     {
999         printf("%d ",stack[i]);
999     }
999     printf("\n");
999 }
999
999 // peek
999 int peek()
999 {
999     if(top<0)
999     {
999         printf("Stack is empty !!!\n");
999         return -1;
999     }
999     else
999     {
999         return stack[top];
999     }
999 }
999
999 // deletion
999 void deletion()
999 {
999     if(top<0)
999     {
999         printf("Stack is empty !!!\n");
999         return;
999     }
999     else
999     {
999         top=top-1;
999     }
999 }
999
999 void insertion(int num)
999 {
999     if(top==49)
999     {
999         printf("\nStack is full !!!\n\n");
999         return;
999     }
999     else
999     {
999         stack[top+1]=num;
999         top=top+1;
999     }
999 }
999
999 // display
999 void display()
999 {
999     int i;
999     for(i=0;i<=top;i++)
999     {
999         printf("%d ",stack[i]);
999     }
999     printf("\n");
999 }
999
999 // peek
999 int peek()
999 {
999     if(top<0)
999     {
999         printf("Stack is empty !!!\n");
999         return -1;
999     }
999     else
999     {
999         return stack[top];
999     }
999 }
999
999 // deletion
999 void deletion()
999 {
999     if(top<0)
999     {
999         printf("Stack is empty !!!\n");
999         return;
999     }
999     else
999     {
999         top=top-1;
999     }
999 }
999
999 void insertion(int num)
999 {
999     if(top==49)
999     {
999         printf("\nStack is full !!!\n\n");
999         return;
999     }
999     else
999     {
999         stack[top+1]=num;
999         top=top+1;
999     }
999 }
999
999 // display
999 void display()
999 {
999     int i;
999     for(i=0;i<=top;i++)
999     {
999         printf("%d ",stack[i]);
999     }
999     printf("\n");
999 }
999
999 // peek
999 int peek()
999 {
999     if(top<0)
999     {
999         printf("Stack is empty !!!\n");
999         return -1;
999     }
999     else
999     {
999         return stack[top];
999     }
999 }
999
999 // deletion
999 void deletion()
999 {
999     if(top<0)
999     {
999         printf("Stack is empty !!!\n");
999         return;
999     }
999     else
999     {
999         top=top-1;
999     }
999 }
999
999 void insertion(int num)
999 {
999     if(top==49)
999     {
999         printf("\nStack is full !!!\n\n");
999         return;
999     }
999     else
999     {
999         stack[top+1]=num;
999         top=top+1;
999     }
999 }
999
999 // display
999 void display()
999 {
999     int i;
999     for(i=0;i<=top;i++)
999     {
999         printf("%d ",stack[i]);
999     }
999     printf("\n");
999 }
999
999 // peek
999 int peek()
999 {
999     if(top<0)
999     {
999         printf("Stack is empty !!!\n");
999         return -1;
999     }
999     else
999     {
999         return stack[top];
999     }
999 }
999
999 // deletion
999 void deletion()
999 {
999     if(top<0)
999     {
999         printf("Stack is empty !!!\n");
999         return;
999     }
999     else
999     {
999         top=top-1;
999     }
999 }
999
999 void insertion(int num)
999 {
999     if(top==49)
999     {
999         printf("\nStack is full !!!\n\n");
999         return;
999     }
999     else
999     {
999         stack[top+1]=num;
999         top=top+1;
999     }
999 }
999
999 // display
999 void display()
999 {
999     int i;
999     for(i=0;i<=top;i++)
999     {
999         printf("%d ",stack[i]);
999     }
999     printf("\n");
999 }
999
999 // peek
999 int peek()
999 {
999     if(top<0)
999     {
999         printf("Stack is empty !!!\n");
999         return -1;
999     }
999     else
999     {
999         return stack[top];
999     }
999 }
999
999 // deletion
999 void deletion()
999 {
999     if(top<0)
999     {
999         printf("Stack is empty !!!\n");
999         return;
999     }
999     else
999     {
999         top=top-1;
999     }
999 }
999
999 void insertion(int num)
999 {
999     if(top==49)
999     {
999         printf("\nStack is full !!!\n\n");
999         return;
999     }
999     else
999
```

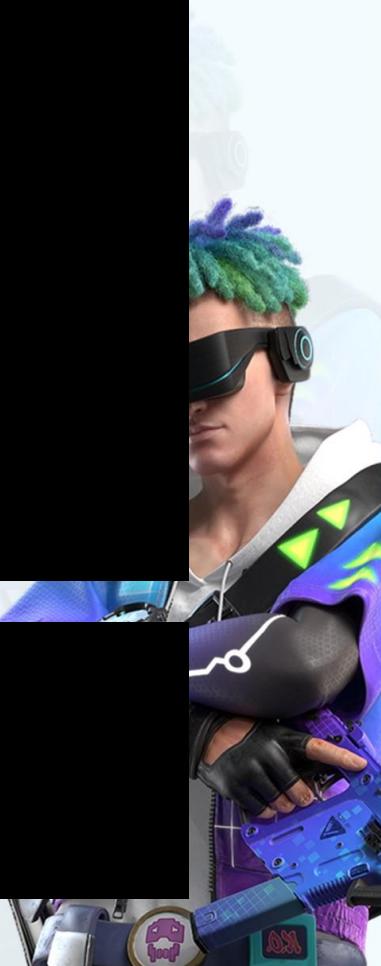
02_queue

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

int queue[50];
int front = -1;
int rear = -1;
```

```
// insertion
void insertion(int num)
{
    if (rear == 49)
    {
        printf("\nQueue is full !!!!\n\n");
        return;
    }
    else if(front== -1 && rear== -1)
    {
        rear++;
        front++;
        queue[rear] = num;
    }
    else
    {
        rear++;
        queue[rear] = num;
    }
}
```

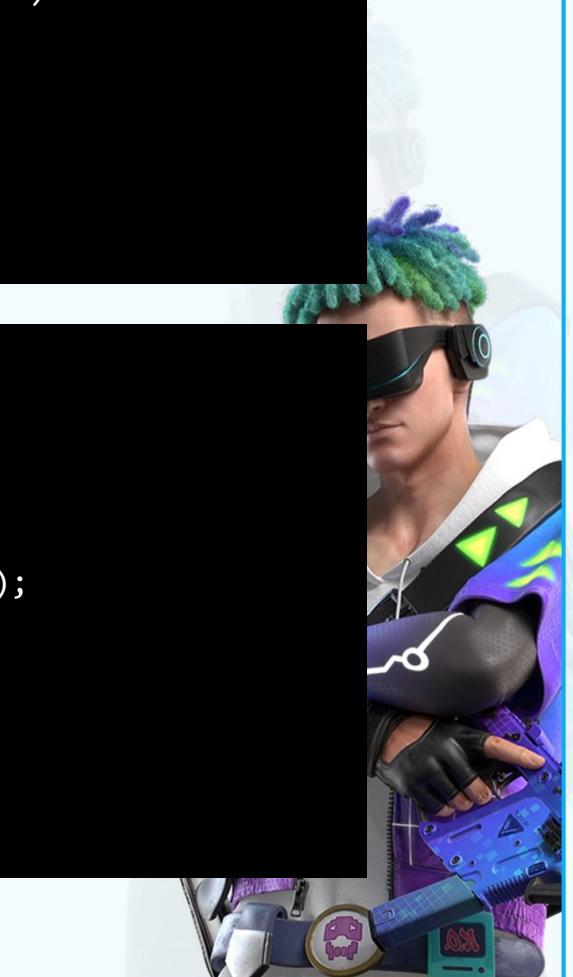
```
// deletion
void deletion()
{
    if (front == -1)
    {
        printf("\nQueue is already empty !!!!\n");
    }
}
```



```
        return;
    }
    else
    {
        int temp = queue[front];
        front++;
        printf("\nElement deleted is %d\n", temp);
    }
}
```

```
// display
void display()
{
    if (front == -1)
    {
        printf("\nQueue is empty !!!!\n");
        return;
    }
    else
    {
        for (int i = front; i <= rear; i++)
        {
            printf("\n%d", queue[i]);
        }
        printf("\n");
    }
}
```

```
// peek
void peek()
{
    if (front == -1)
    {
        printf("\nQueue is empty !!!!\n");
        return;
    }
    else
    {
        printf("\n%d\n", queue[front]);
    }
}
```



```
}
```

```
int main()
{
    char ch[5] = "yes";
    do
    {
        int choice;
        printf("\nSelect your choice:\n1.Insertion\n2.Deletion\n3.Display\n4.Peek\n5.exit\n");
        scanf("%d", &choice);
        switch (choice)
        {
            case 1:
                int num;
                printf("Enter the number to insert: ");
                scanf("%d", &num);
                insertion(num);
                break;

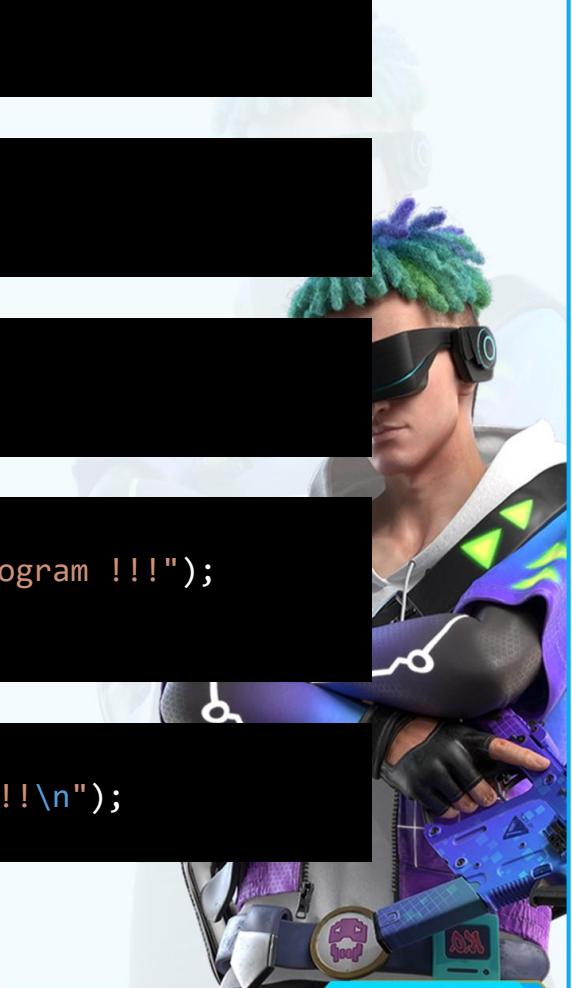
            case 2:
                deletion();
                break;

            case 3:
                display();
                break;

            case 4:
                peek();
                break;

            case 5:
                printf("\nTerminating the program !!!");
                exit(1);
                break;

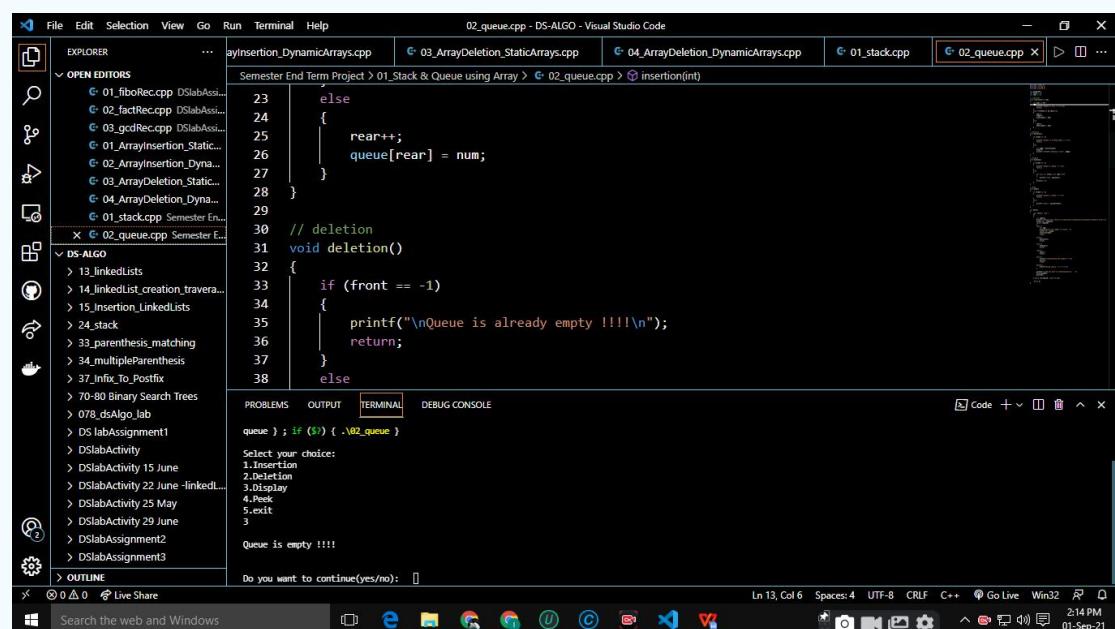
            default:
                printf("Wrong choice !!!!!!!\n");
        }
    }
}
```



```
    printf("\n\nDo you want to continue(yes/no):    ");
    fflush(stdin);
    gets(ch);
```

```
} while (strcmp(ch, "yes") == 0);
```

```
return 0;
}
```



```
else
{
    rear++;
    queue[rear] = num;
}

// deletion
void deletion()
{
    if (front == -1)
    {
        printf("\nQueue is already empty !!!!\n");
        return;
    }
    else
    {
        printf("\nQueue is empty !!!!\n");
        return;
    }
}

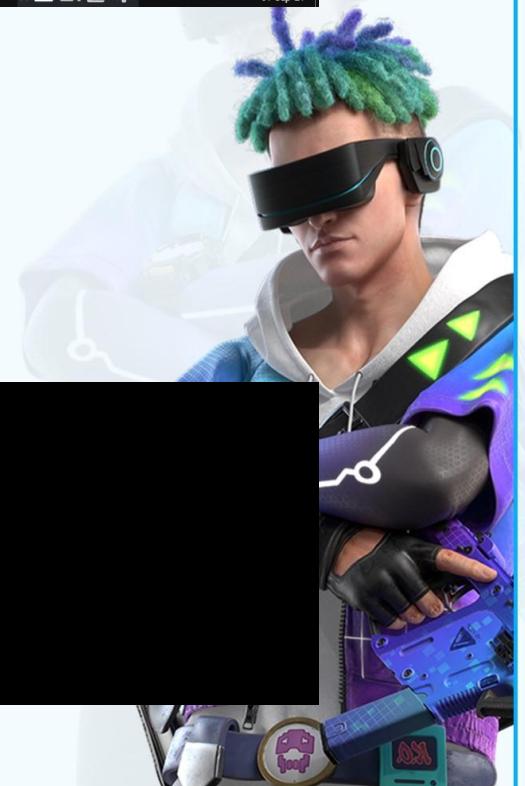
Select your choice:
1.Insertion
2.Deletion
3.Display
4.Peek
5.Exit
3

Do you want to continue(yes/no):
```

03_circularQueue

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#define max 50

int queue[max];
int front = -1;
```



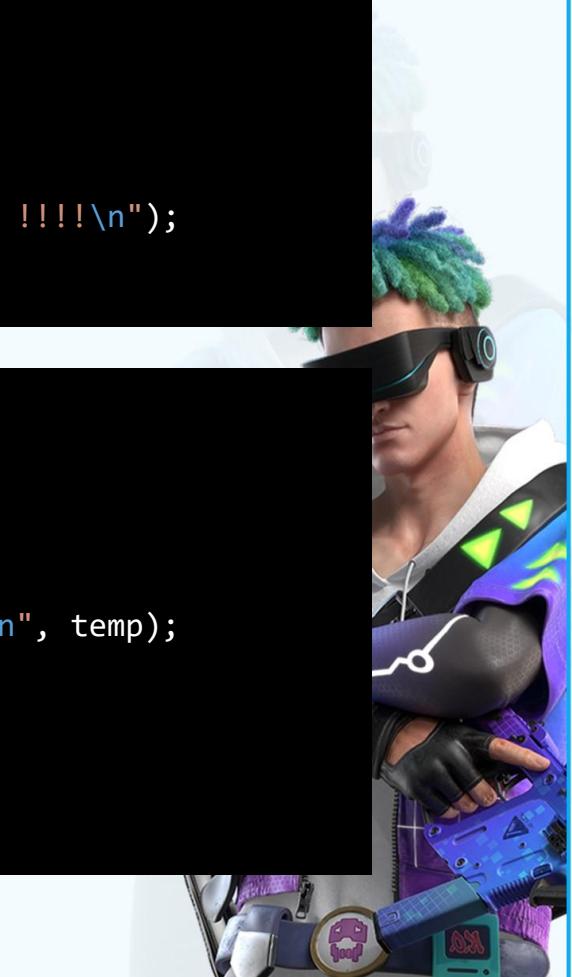
```
int rear = -1;

// insertion
void insertion(int num)
{
    if ((rear+1)%(max-1) == front)
    {
        printf("\nQueue is full !!!!\n\n");
        return;
    }
    else if (front == -1)
    {
        front++;
    }

    rear = (rear+1)%(max-1);
    queue[rear] = num;
}
```

```
// deletion
void deletion()
{
    if (front == -1)
    {
        printf("\nQueue is already empty !!!!\n");
        return;
    }
```

```
else if(front==rear)
{
    int temp = queue[front];
    front=-1;
    rear=-1;
    printf("\nElement deleted is %d\n", temp);
}
else
{
    int temp = queue[front];
    front++;
```



```
        printf("\nElement deleted is %d\n", temp);
    }
}
```

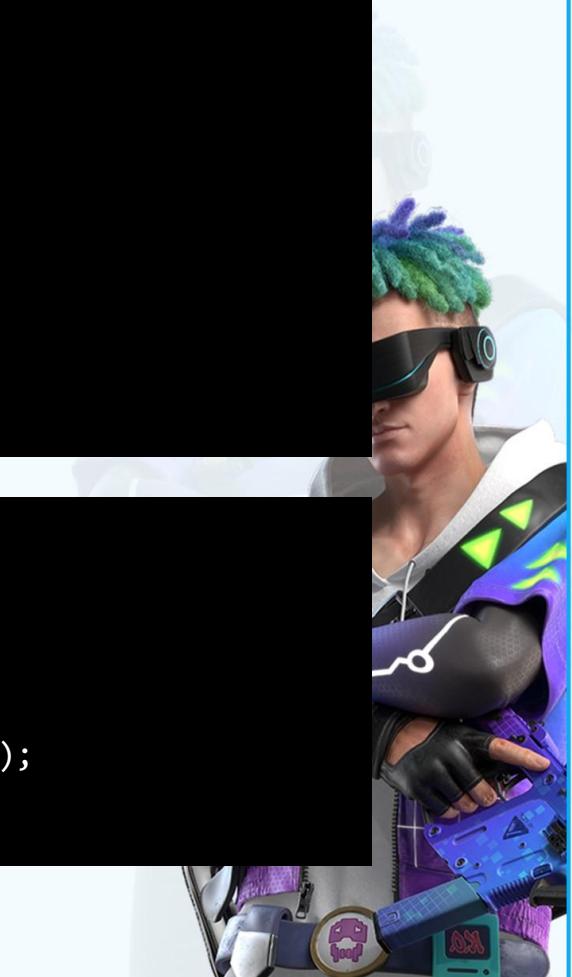
```
// display
void display()
{
    printf("\nFront = %d\nRear = %d\n", front, rear);
```

```
    if (front == -1)
    {
        printf("\nQueue is empty !!!!\n");
        return;
    }
```

```
    else if(front==rear)
    {
        printf("\n%d\n", queue[front]);
    }
```

```
else
{
    int i = front;
    while(i!=rear)
    {
        printf("\n%d", queue[i]);
        i=(i+1)% (max-1);
    }
    printf("\n%d\n", queue[i]);
}
```

```
// peek
void peek()
{
    if (front == -1)
    {
        printf("\nQueue is empty !!!!\n");
        return;
    }
```



```
        else
        {
            printf("\n%d\n", queue[front]);
        }
    }
```

```
int main()
{
    char ch[5] = "yes";
    do
    {
        int choice;
        printf("\nSelect your choice:\n1.Insertion\n2.Deletion\n3.Display\n4.Peek\n5.exit\n");
        scanf("%d", &choice);
        switch (choice)
        {
            case 1:
                int num;
                printf("Enter the number to insert: ");
                scanf("%d", &num);
                insertion(num);
                break;

            case 2:
                deletion();
                break;

            case 3:
                display();
                break;

            case 4:
                peek();
                break;

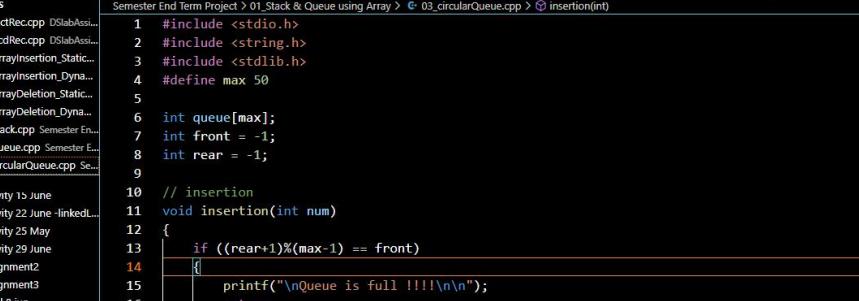
            case 5:
                printf("\nTerminating the program !!!");
                exit(1);
                break;
        }
    }
}
```

```
default:  
    printf("Wrong choice !!!!!!!\n");  
}
```

```
printf("\n\nDo you want to continue(yes/no):    ");
fflush(stdin);
gets(ch);
```

```
} while (strcmp(ch, "yes") == 0);
```

```
    return 0;  
}
```



File Edit Selection View Go Run Terminal Help 03_circularQueue.cpp - DS-ALGO - Visual Studio Code

EXPLORER ...

OPEN EDITORS

- 02_factRec.cpp DSlabAssignment1
- 03_gdRec.cpp DSlabAssignment1
- 01_ArrayInsertion_Static...
- 02_ArrayInsertion_Dyna...
- 03_ArrayDeletion_Static...
- 04_ArrayDeletion_Dyna...
- 01_stack.cpp Semester End Project
- 02_queue.cpp Semester End Project
- 03_circularQueue.cpp Semester End Project

DS-ALGO

- > DSlabActivity 15 June
- > DSlabActivity 22 June - linked...
- > DSlabActivity 25 May
- > DSlabActivity 29 June
- > DSlabAssignment2
- > DSlabAssignment3
- > lab manual 8 jun
- > linked list polynomial
- > other programs

Semester End Term Project | 0...

- 01_stack.cpp
- 01_stack.exe
- 02_queue.cpp
- 02_queue.exe
- 03_circularQueue.cpp
- 03_circularQueue.exe
- temp.cpp

> OUTLINE

03_ArrayDeletion_StaticArrays.cpp 04_ArrayDeletion_DynamicArrays.cpp 01_stack.cpp 02_queue.cpp 03_circularQueue.cpp

```
1 #include <stdio.h>
2 #include <string.h>
3 #include <stdlib.h>
4 #define max 50
5
6 int queue[max];
7 int front = -1;
8 int rear = -1;
9
10 // insertion
11 void insertion(int num)
12 {
13     if ((rear+1)% (max-1) == front)
14     {
15         printf("\nQueue is full !!!!\n\n");
16         return;
17     }
18 }
19
20 void display()
21 {
22     int i;
23     for (i = front+1; i < rear; i++)
24     {
25         printf("%d ", queue[i]);
26     }
27 }
```

PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE

```
p -> 03_circularQueue ; if (g) { .03_circularQueue }
```

Select your choice:

- 1.Insertion
- 2.Deletion
- 3.Display
- 4.Peek
- 5.exit
- 4

Queue is empty !!!!

Do you want to continue(yes/no):

Ln 14, Col 6 Spaces: 4 UTF-8 CRLF C++ Go Live Win32

2:16 PM 01-Sep-21

Search the web and Windows



Lab Assignment 6

01_queueOperations

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

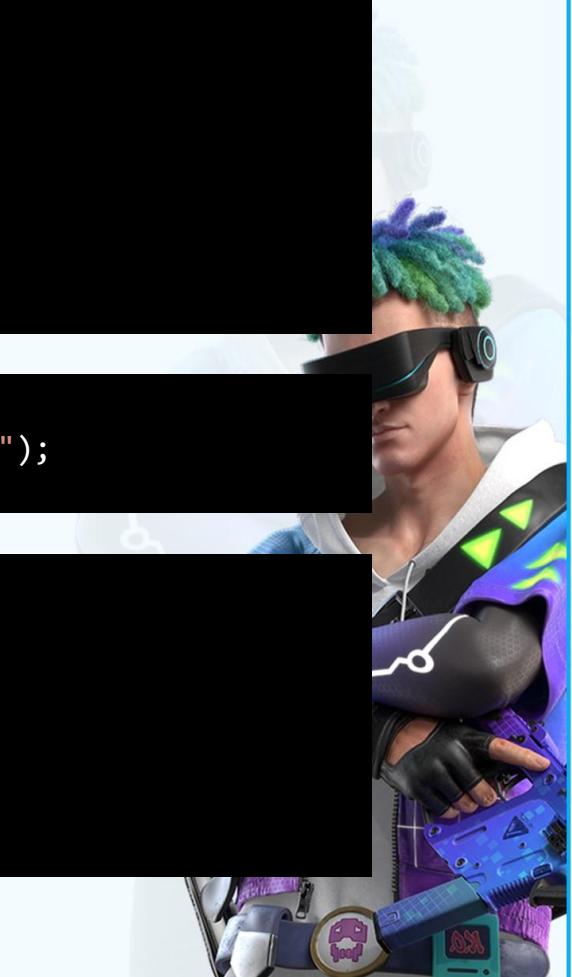
int queue[50];
int front = -1;
int rear = -1;
```

```
// insertion
void insertion(int item)
{
    if(rear<49)
    {
        if(front== -1)
            front++;

        rear++;
        queue[rear]=item;
    }

    else
        printf("\nThe queue is full !!!!");
}
```

```
// deletion
void deletion()
{
    int item;
    if (front != -1)
    {
        item=queue[front];
```



```

        if(front==rear)
        {
            front=-1;
            rear=-1;
        }

    else
        front++;



        printf("\nNumber deleted is %d\n", item);
    }
else
{
    printf("\nQueue is already empty !!!!\n");
}
}

```

```

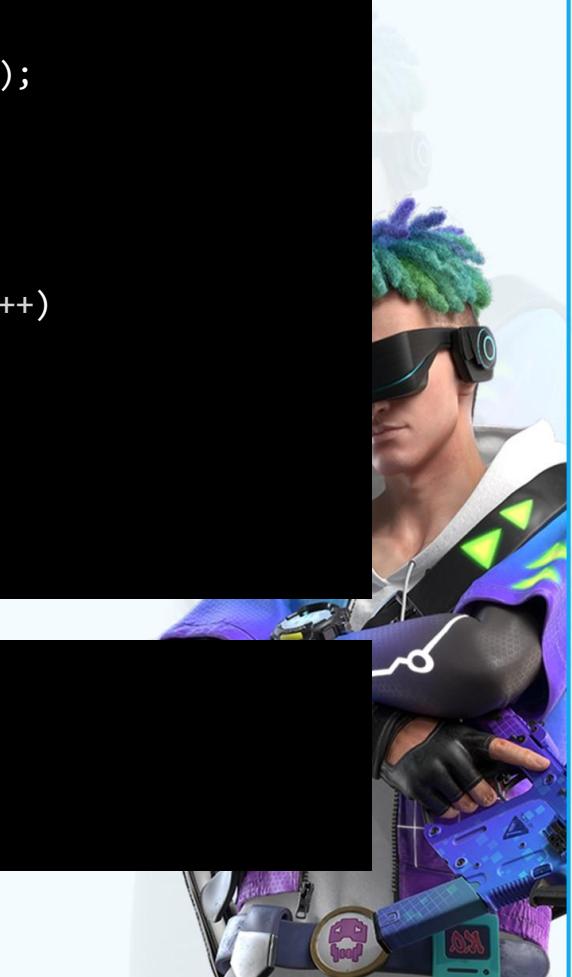
// display
void display()
{
    if (front == -1)
    {
        printf("\nQueue is empty !!!!\n");
        return;
    }
    else
    {
        for (int i = front; i <= rear; i++)
        {
            printf("\n%d", queue[i]);
        }
        printf("\n");
    }
}

```

```

// peek
void peek()
{
    if (front == -1)
    {

```



```
        printf("\nQueue is empty !!!!\n");
        return;
    }
    else
    {
        printf("\n%d\n", queue[front]);
    }
}
```

```
int main()
{
    char ch[5] = "yes";
    do
    {
        int choice;
        printf("\nSelect your choice:\n1.Insertion\n2.Deletion\n3.Display\n4.Peek\n5.exit\n");
        scanf("%d", &choice);
        switch (choice)
        {
            case 1:
                int num;
                printf("Enter the number to insert: ");
                scanf("%d", &num);
                insertion(num);
                break;

            case 2:
                deletion();
                break;

            case 3:
                display();
                break;

            case 4:
                peek();
                break;

            case 5:
                break;
        }
    }
}
```

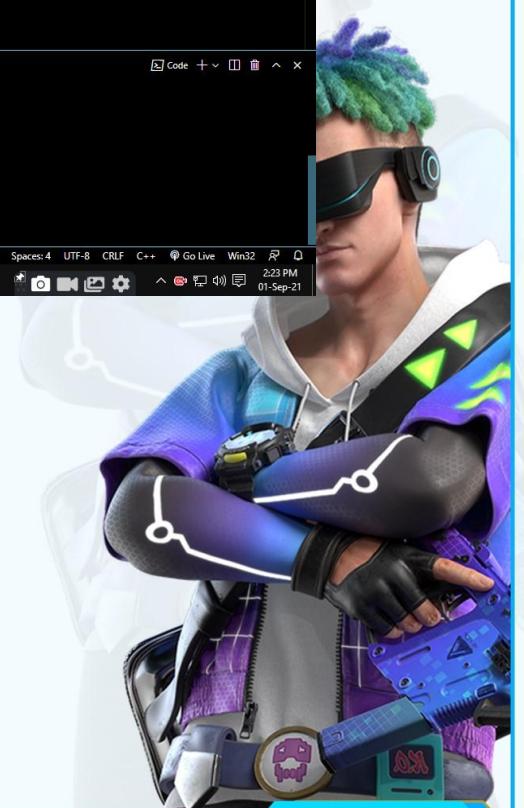
```
        printf("\nTerminating the program !!!");
        exit(1);
        break;
```

```
    default:
        printf("Wrong choice !!!!!!!\n");
    }
```

```
printf("\n\nDo you want to continue(yes/no):   ");
fflush(stdin);
gets(ch);
```

```
} while (strcmp(ch, "yes") == 0);
```

```
return 0;
}
```



File Edit Selection View Go Run Terminal Help 01_queueOperations.cpp - DS-ALGO - Visual Studio Code

OPEN EDITORS

- 01_queueOperations.cpp
- 04_ArrayDeletion_DynamicArrays.cpp
- 01_stack.cpp
- 02_queue.cpp
- 03_circularQueue.cpp
- 01_queueOperations.cpp

DSLabActivity 25 May > 01.queueOperations.cpp > ...

```
1 #include <stdio.h>
2 #include <string.h>
3 #include <stdlib.h>
4
5 int queue[50];
6 int front = -1;
7 int rear = -1;
8
9 // insertion
10 void insertion(int item)
11 {
12     if(rear<49)
13     {
14         if(front==-1)
15             front++;
16     }
17 }
```

PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE

```
5.exit
1
Enter the number to insert: 4

Do you want to continue(yes/no): yes

Select your choice:
1.Insertion
2.Deletion
3.Display
4.Peek
5.Exit
```

Ln 9, Col 13 Spaces: 4 UTF-8 CRLF C++ Go Live Win32 2:23 PM 01-Sep-21

Search the web and Windows

Lab Assignment 7

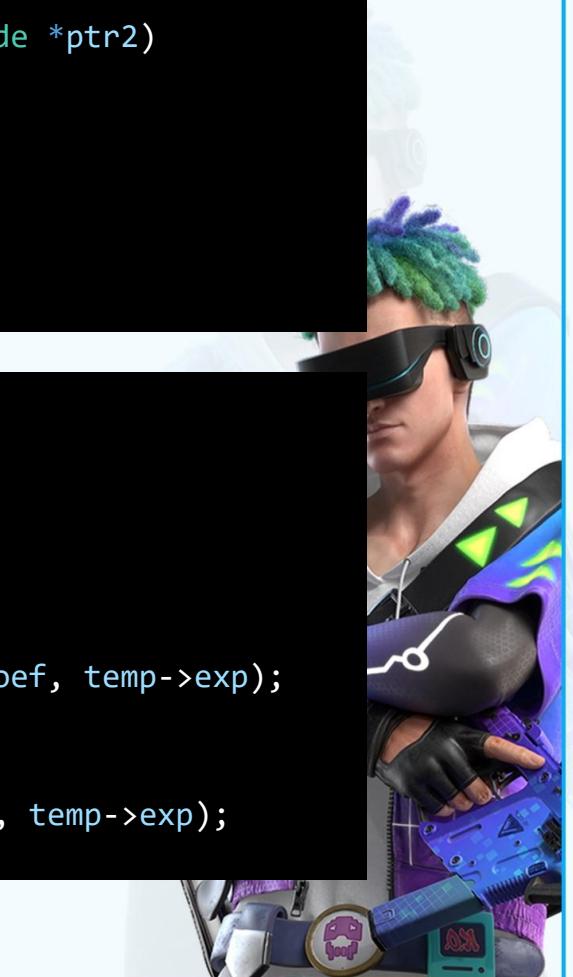
01_linkedList_polynomial

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

struct node
{
    int exp;
    float coef;
    struct node *next;
};

// function append
// appends ptr1 to ptr2
void append(struct node *ptr1, struct node *ptr2)
{
    struct node *ptr = ptr2;
    while (ptr->next != NULL)
        ptr = ptr->next;
    ptr->next = ptr1;
}

// function for traversal
void traversal(struct node *head)
{
    struct node *temp = head;
    while (temp->next != NULL)
    {
        printf("%4.1f(x)^%d + ", temp->coef, temp->exp);
        temp = temp->next;
    }
    printf("%4.1f(x)^%d\n\n", temp->coef, temp->exp);
}
```



```

// function for insertion
struct node *insertion(struct node *head, int ex, float cof)
{
    struct node *temp = (struct node *)malloc(sizeof(struct
node));
    // printf("%4.2fx^%d\n", cof, ex);
    temp->coef = cof;
    temp->exp = ex;
    temp->next = NULL;
    // printf("%4.2fx^%d\n", temp->coef, temp->exp);
    if (head == NULL)
        head = temp;
    else
    {
        struct node *ptr = head;
        while (ptr->next != NULL)
            ptr = ptr->next;

        ptr->next = temp;
    }
    return head;
}

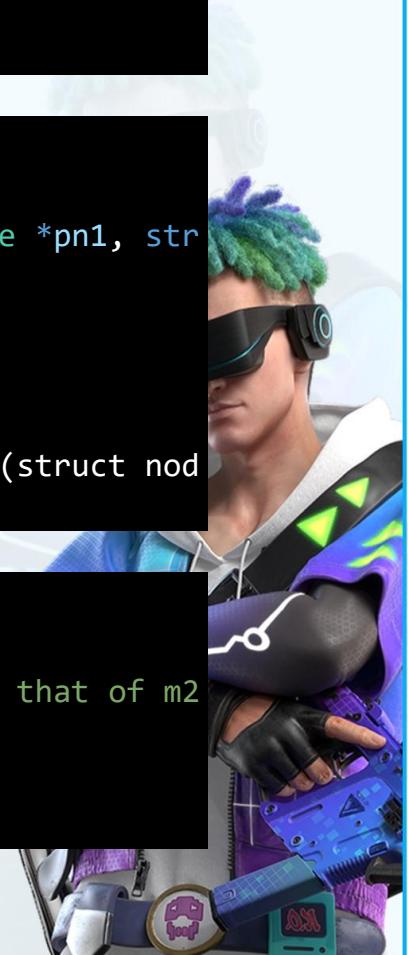
```

```

// function to add two polynomials
// pn1 + pn2 = pn3
struct node *addPoly(struct node *pn3, struct node *pn1, str
uct node *pn2)
{
    struct node *m1 = pn1;
    struct node *m2 = pn2;
    struct node *m3 = (struct node*)malloc(sizeof(struct nod
e));

    while((m1!=NULL) || (m2!=NULL))
    {
        // when power of m1's term is higher than that of m2
's
        while(m1->exp > m2->exp)
        {

```



```
    m3->exp = m1->exp;
    m3->coef = m1->coef;
    append(m3, pn3);
    m1 = m1->next;
}
```

```
// when power of m2's term is higher than that of m1
's
while (m1->exp < m2->exp)
{
    m3->exp = m2->exp;
    m3->coef = m2->coef;
    append(m3, pn3);
    m2 = m2->next;
}
```

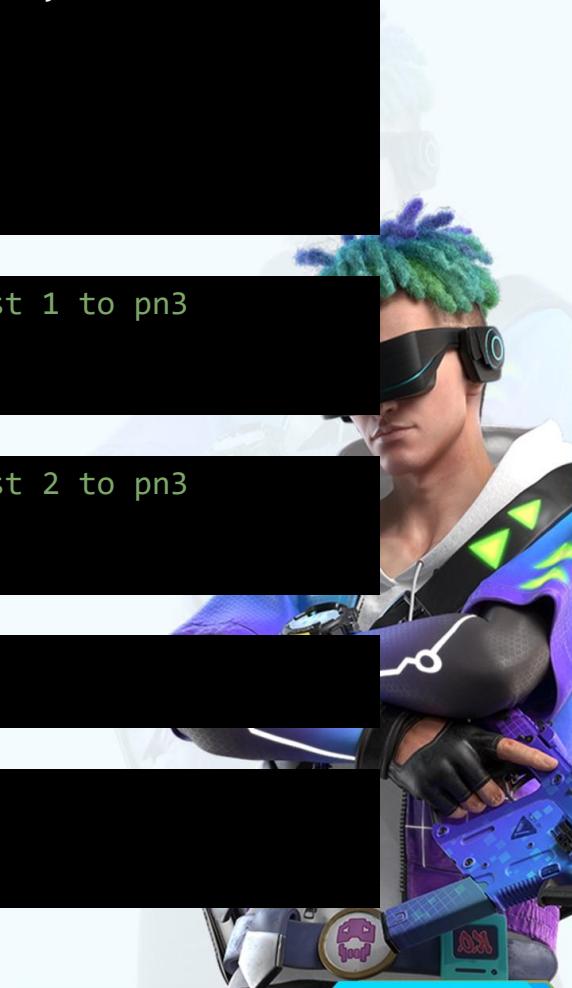
```
// when power of m2's term is equal to that of m1's
while (m1->exp == m2->exp)
{
    m3->exp = m2->exp;
    m3->coef = m2->coef + m1->coef;
    append(m3, pn3);
    m2 = m2->next;
    m1 = m1->next;
}
}
```

```
// if list 2 exhausts, appending list 1 to pn3
if(m1!=NULL)
    append(m1, pn3);
```

```
// if list 1 exhausts, appending list 2 to pn3
else
    append(m2, pn3);
```

```
return pn3;
}
```

```
int main()
{
    char ch = 'y';
```



```
struct node *pn1 = NULL, *pn2 = NULL, *pn3 = NULL;

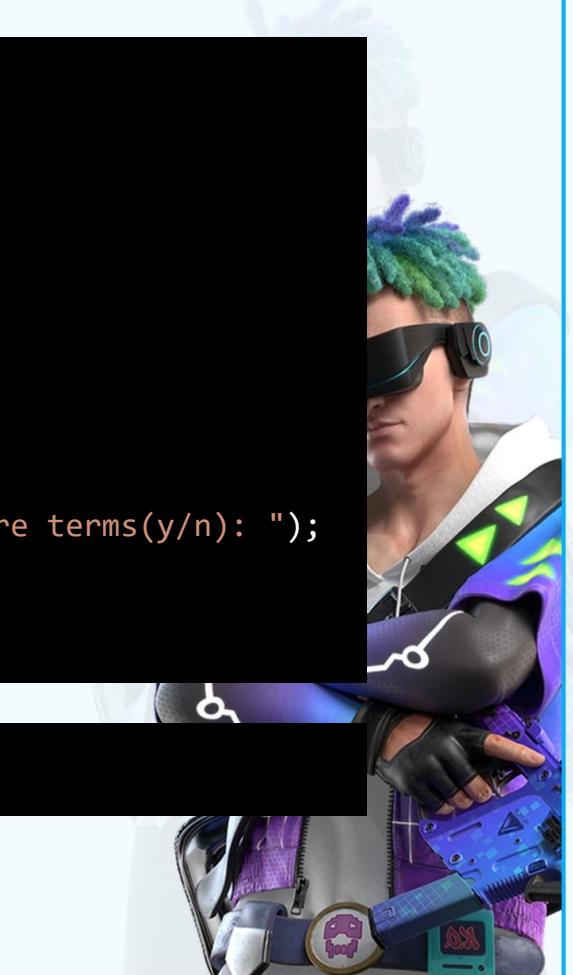
printf("Polynomial 1: \n");
while (ch == 'y')
{
    int ex;
    float cof;
    printf("Enter Coefficient : ");
    scanf("%f", &cof);
    printf("Enter Exponent : ");
    scanf("%d", &ex);
    pn1 = insertion(pn1, ex, cof);
    printf("Do you want to insert more terms(y/n): ");
    fflush(stdin);
    scanf("%c", &ch);
}
```

```
printf("The polynomial 1 is : ");
traversal(pn1);
```

```
ch = 'y'; // reinitialising y = 'y'
```

```
printf("Polynomial 2: \n");
while (ch == 'y')
{
    int ex;
    float cof;
    printf("Enter Coefficient : ");
    scanf("%f", &cof);
    printf("Enter Exponent : ");
    scanf("%d", &ex);
    pn2 = insertion(pn2, ex, cof);
    printf("Do you want to insert more terms(y/n): ");
    fflush(stdin);
    scanf("%c", &ch);
}
```

```
printf("The polynomial 2 is : ");
traversal(pn2);
```



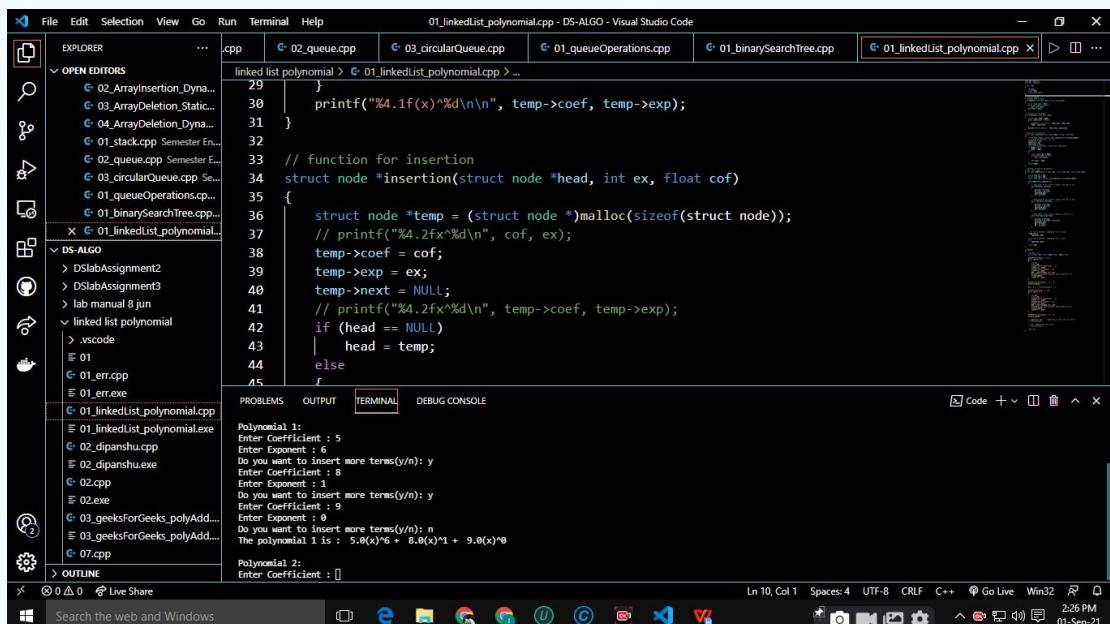
```

        // append(pn1,pn2); // append pn1 to pn2, just for test
    :)
    // traversal(pn2);

    // pn3 = addPoly(pn3, pn2, pn1);
    // traversal(pn3);

    return 0;
}

```



```

File Edit Selection View Go Run Terminal Help
01_LinkedList_polynomial.cpp - DS-ALGO - Visual Studio Code
EXPLORER OPEN EDITORS .cpp 02_queue.cpp 03_circularQueue.cpp 01_queueOperations.cpp 01_binarySearchTree.cpp 01_LinkedList_polynomial.cpp
OPEN EDITORS
linked list polynomial > 01_LinkedList_polynomial.cpp ...
29 }
30     printf("%4.1f(x)^%d\n\n", temp->coef, temp->exp);
31 }
32
33 // function for insertion
34 struct node *insertion(struct node *head, int ex, float cof)
35 {
36     struct node *temp = (struct node *)malloc(sizeof(struct node));
37     // printf("%4.2fx^%d\n", cof, ex);
38     temp->coef = cof;
39     temp->exp = ex;
40     temp->next = NULL;
41     // printf("%4.2fx^%d\n", temp->coef, temp->exp);
42     if (head == NULL)
43     {
44         head = temp;
45     }
        ...
PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE
Code + v ^ x
Polynomial 1:
Enter Coefficient : 5
Enter Exponent : 6
Do you want to insert more terms(y/n): y
Enter Coefficient : 8
Enter Exponent : 5
Do you want to insert more terms(y/n): y
Enter Coefficient : 9
Enter Exponent : 0
Do you want to insert more terms(y/n): n
The polynomial 1 is : 5.0(x)^6 + 8.0(x)^5 + 9.0(x)^0
Polynomial 2:
Enter Coefficient : 0
Ln 10, Col 1 Spaces: 4 UTF-8 CRLF C++ Go Live Win32 2:26 PM 01-Sep-21

```

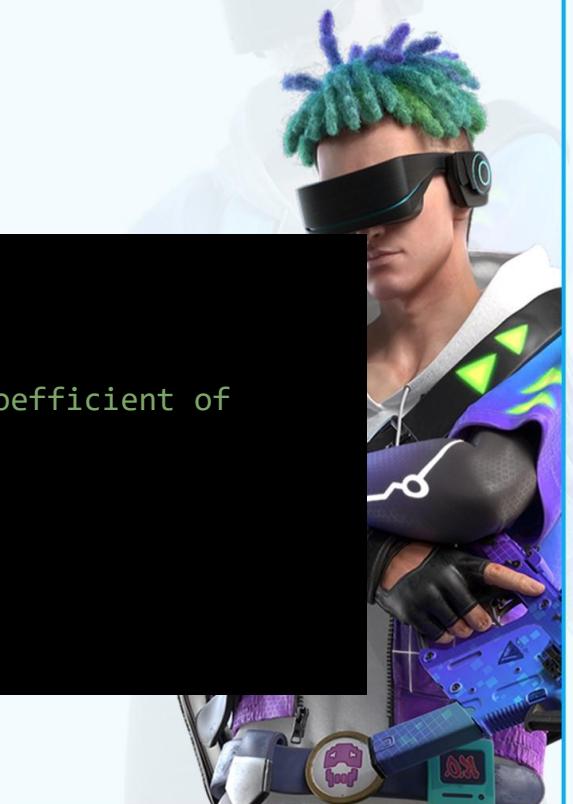
03_geeksForGeeks_polyAdd

```

#include <bits/stdc++.h>
using namespace std;

// Node structure containing power and coefficient of
// variable
struct Node
{
    int coeff;
    int pow;
    struct Node *next;
}

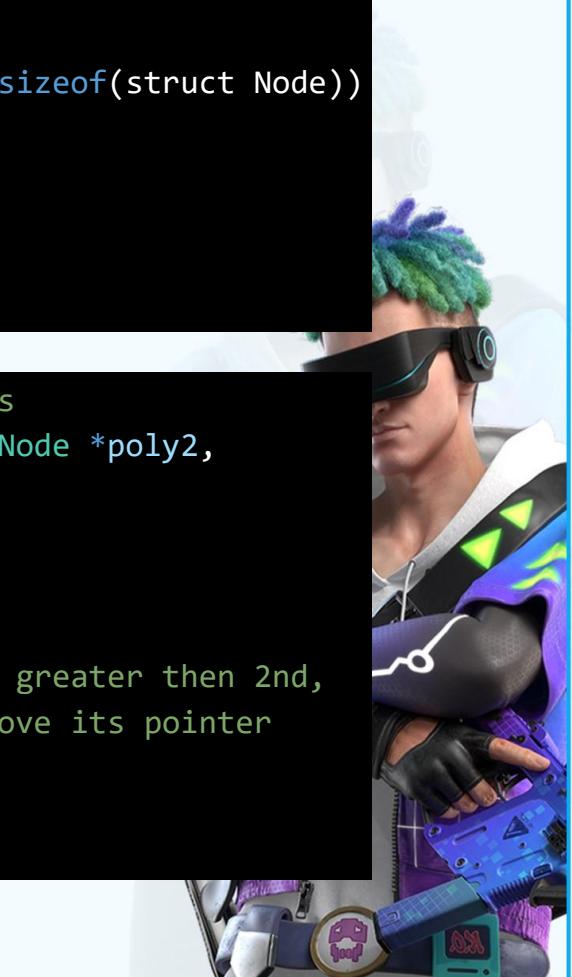
```



```
};
```

```
// Function to create new node
void create_node(int x, int y, struct Node **temp)
{
    struct Node *r, *z;
    z = *temp;
    if (z == NULL)
    {
        r = (struct Node *)malloc(sizeof(struct Node));
        r->coeff = x;
        r->pow = y;
        *temp = r;
        r->next = (struct Node *)malloc(sizeof(struct Node))
    ;
        r = r->next;
        r->next = NULL;
    }
    else
    {
        r->coeff = x;
        r->pow = y;
        r->next = (struct Node *)malloc(sizeof(struct Node))
    ;
        r = r->next;
        r->next = NULL;
    }
}
```

```
// Function Adding two polynomial numbers
void polyadd(struct Node *poly1, struct Node *poly2,
             struct Node *poly)
{
    while (poly1->next && poly2->next)
    {
        // If power of 1st polynomial is greater then 2nd,
        // then store 1st as it is and move its pointer
        if (poly1->pow > poly2->pow)
        {
            poly->pow = poly1->pow;
```



```

        poly->coeff = poly1->coeff;
        poly1 = poly1->next;
    }

    // If power of 2nd polynomial is greater then 1st,
    // then store 2nd as it is and move its pointer
    else if (poly1->pow < poly2->pow)
    {
        poly->pow = poly2->pow;
        poly->coeff = poly2->coeff;
        poly2 = poly2->next;
    }

    // If power of both polynomial numbers is same then
    // add their coefficients
    else
    {
        poly->pow = poly1->pow;
        poly->coeff = poly1->coeff + poly2->coeff;
        poly1 = poly1->next;
        poly2 = poly2->next;
    }

    // Dynamically create new node
    poly->next = (struct Node *)malloc(sizeof(struct Node));
    poly = poly->next;
    poly->next = NULL;
}

while (poly1->next || poly2->next)
{
    if (poly1->next)
    {
        poly->pow = poly1->pow;
        poly->coeff = poly1->coeff;
        poly1 = poly1->next;
    }
    if (poly2->next)
    {
        poly->pow = poly2->pow;
    }
}

```



```

        poly->coeff = poly2->coeff;
        poly2 = poly2->next;
    }
    poly->next = (struct Node *)malloc(sizeof(struct Node));
    poly = poly->next;
    poly->next = NULL;
}
}

```

```

// Display Linked list
void show(struct Node *node)
{
    while (node->next != NULL)
    {
        printf("%dx^%d", node->coeff, node->pow);
        node = node->next;
        if (node->coeff >= 0)
        {
            if (node->next != NULL)
                printf("+");
        }
    }
    printf("\n");
}

```

```

// Driver code
int main()
{
    struct Node *poly1 = NULL, *poly2 = NULL, *poly = NULL;

    // Create first list of 5x^2 + 4x^1 + 2x^0
    create_node(5, 2, &poly1);
    create_node(4, 1, &poly1);
    create_node(2, 0, &poly1);

    // Create second list of -5x^1 - 5x^0
    create_node(-5, 1, &poly2);
    create_node(-5, 0, &poly2);
}

```

```
    printf("1st Number: ");
    show(poly1);
```

```
    printf("2nd Number: ");
    show(poly2);
```

```
    poly = (struct Node *)malloc(sizeof(struct Node));
```

```
    // Function add two polynomial numbers
    polyadd(poly1, poly2, poly);
```

```
    // Display resultant List
    printf("Added polynomial: ");
    show(poly);
```

```
    return 0;
}
```



A screenshot of Visual Studio Code showing the code for polynomial addition. The code uses a linked list structure to represent polynomials. It includes functions for creating nodes, printing polynomials, and adding them. The code is in C++ and is part of a larger project structure.

```
File Edit Selection View Go Run Terminal Help
03_geeksForGeeks_polyAdd.cpp - DS-ALGO - Visual Studio Code
EXPLORER OPEN EDITORS circularQueue.cpp 01_queueOperations.cpp 01_binarySearchTree.cpp 01_linkedList_polynomial.cpp 03_geeksForGeeks_polyAdd.cpp ...
linked list polynomial > 03_geeksForGeeks_polyAdd.cpp > ...
1 // C++ program for addition of two polynomials
2 // using Linked Lists
3 #include <bits/stdc++.h>
4 using namespace std;
5
6 // Node structure containing power and coefficient of
7 // variable
8 struct Node
9 {
10     int coeff;
11     int pow;
12     struct Node *next;
13 };
14
15 // Function to create new node
16 void create_node(int x, int y, struct Node **temp)
{
    *temp = new Node;
    (*temp)->coeff = x;
    (*temp)->pow = y;
    (*temp)->next = NULL;
}
17
18 // Function to print polynomial
19 void show(struct Node *poly)
{
    if (poly == NULL)
        return;
    cout << poly->coeff << "x^" << poly->pow << " + ";
    show(poly->next);
}
20
21 // Function to add two polynomials
22 void polyadd(struct Node *poly1, struct Node *poly2, struct Node **poly)
{
    if (poly1 == NULL)
        *poly = poly2;
    else if (poly2 == NULL)
        *poly = poly1;
    else
        if (poly1->pow > poly2->pow)
            polyadd(poly2, poly1, poly);
        else
            if (poly1->pow == poly2->pow)
                poly1->coeff = poly1->coeff + poly2->coeff;
                polyadd(poly1->next, poly2, poly);
            else
                polyadd(poly1, poly2->next, poly);
}
23
24 // Driver program
25 int main()
26 {
    struct Node *poly1, *poly2, *poly;
    int x, y;
    cout << "1st Number: ";
    cin << x << " " << y << endl;
    cout << "2nd Number: ";
    cin << x << " " << y << endl;
    create_node(x, y, &poly1);
    cout << "Added polynomial: ";
    show(poly1);
    return 0;
}

```

Lab Assignment 8

01_sorting

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

void display(int a[], int n);
void bubble_sort(int a[], int n);
void selection_sort(int a[], int n);
void insertion_sort(int a[], int n);

int main()
{
    int n, choice, i;
    char ch[20];
    printf("Enter no. of elements u want to sort : ");
    scanf("%d", &n);
    int arr[n];
    for (i = 0; i < n; i++)
    {
        printf("Enter %d Element : ", i + 1);
        scanf("%d", &arr[i]);
    }
    printf("Please select any option Given Below for Sorting
: \n");

    while (1)
    {

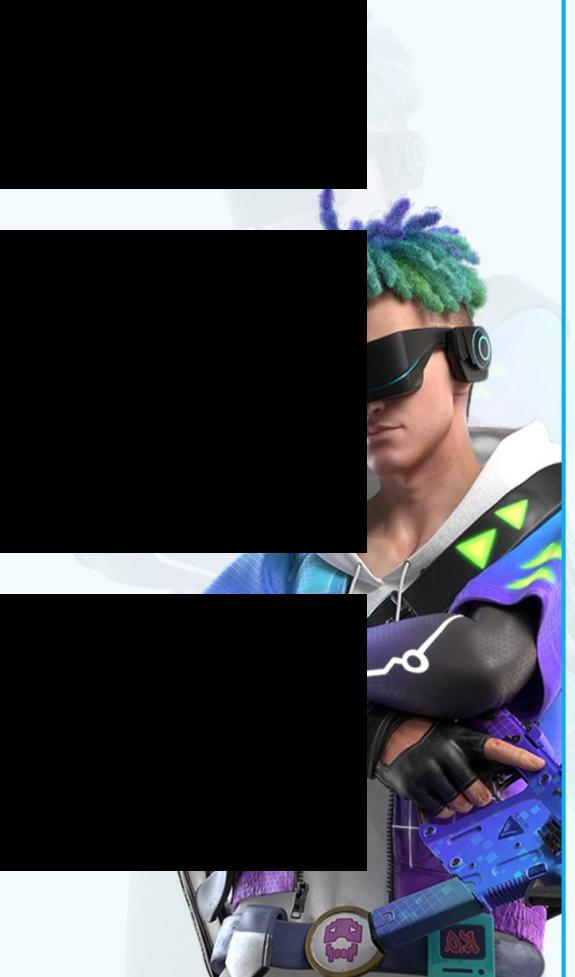
        printf("\n1. Bubble Sort\n2. Selection Sort\n3. Inse
rtion Sort\n4. Display Array.\n5. Exit the Program.\n");
        printf("\nEnter your Choice : ");
        scanf("%d", &choice);
    }
}
```

```
switch (choice)
{
case 1:
    bubble_sort(arr, n);
    break;
case 2:
    selection_sort(arr, n);
    break;
case 3:
    insertion_sort(arr, n);
    break;
case 4:
    display(arr, n);
    break;
```

```
case 5:
    return 0;
default:
    printf("\nPlease Select only 1-5 option ----\n");
}
}
return 0;
}
```

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

```
void bubble_sort(int arr[], int n)
{
    int i, j, temp;
    for (i = 0; i < n; i++)
    {
        for (j = 0; j < n - i - 1; j++)
```



```

    {
        if (arr[j] > arr[j + 1])
        {
            temp = arr[j];
            arr[j] = arr[j + 1];
            arr[j + 1] = temp;
        }
    }
    printf("After Bubble sort Elements are : ");
    display(arr, n);
}

```

```

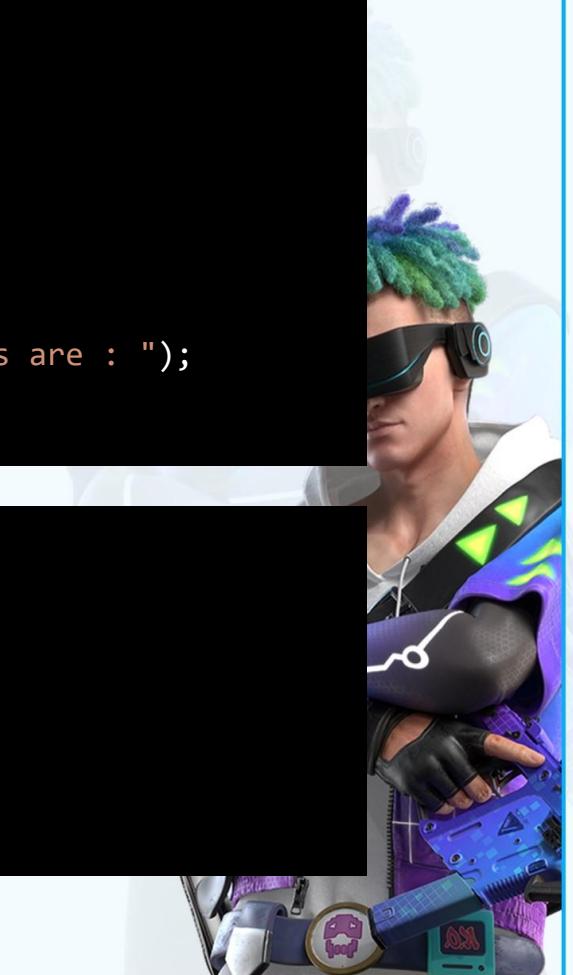
void selection_sort(int arr[], int n)
{
    int i, j, temp;
    for (i = 0; i < n - 1; i++)
    {
        for (j = i + 1; j < n; j++)
        {
            if (arr[i] > arr[j])
            {
                temp = arr[i];
                arr[i] = arr[j];
                arr[j] = temp;
            }
        }
    }
    printf("After Selection sort Elements are : ");
    display(arr, n);
}

```

```

void insertion_sort(int arr[], int n)
{
    int i, j, min;
    for (i = 1; i < n; i++)
    {
        min = arr[i];
        j = i - 1;
        while (min < arr[j] && j >= 0)

```



```

    {
        arr[j + 1] = arr[j];
        j = j - 1;
    }
    arr[j + 1] = min;
}
printf("After Insertion sort Elements are : ");
display(arr, n);
}

```



A screenshot of the Visual Studio Code interface. The terminal window shows a PowerShell session running on a Windows 10 system. The user is executing a C++ program named '01_sorting.cpp' which performs an insertion sort on an array of integers. The program outputs the sorted array and asks for user input to choose a sorting algorithm. The user has selected 'Insertion Sort' three times, and the program has sorted the array each time. The terminal also shows the system tray and taskbar at the bottom.

```

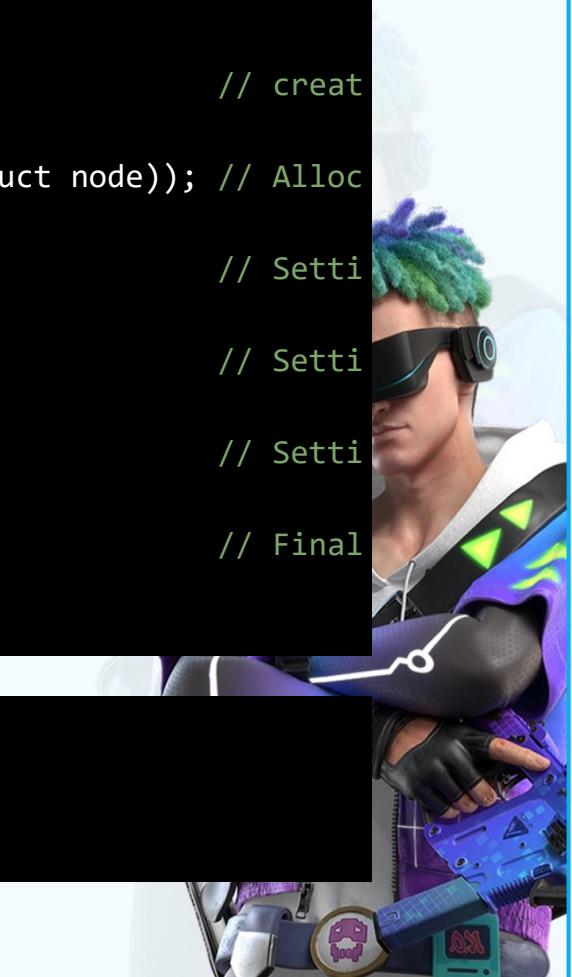
File Edit Selection View Go Run Terminal Help
01_sorting.cpp - programs of C - Visual Studio Code
EXPLORER PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE
OPEN EDITORS
  01_Welcome
  05_TreeTraversal.cpp gunu
  01_sorting.exe Divya
  01_sorting.cpp Divya
PROGRAMS OF C
  > vscode
  > Babita
  > Dipanshu
  > Divya
  01_sorting.cpp
  01_sorting.exe
  > function pointer
  > gurpreet
  > 01_Linked_lists.cpp
  01_Linked_lists.exe
  02_implementation_of_stack...
  02_implementation_of_stack...
  03_stackComp.cpp
  03_stackComp.exe
  04_stackForSeparateNodes.c...
  04_stackForSeparateNodes.e...
  05_TreeTraversal.cpp
  05_TreeTraversal.exe
  06_EndTermPractical.cpp
  06_EndTermPractical.exe
  07_endTermPractical.cpp
  07_endTermPractical.exe
  > OUTLINE
PROBLEMS
  01_sorting.exe
  01_sorting.cpp
  01_Linked_lists.cpp
  01_Linked_lists.exe
  02_implementation_of_stack...
  02_implementation_of_stack...
  03_stackComp.cpp
  03_stackComp.exe
  04_stackForSeparateNodes.c...
  04_stackForSeparateNodes.e...
  05_TreeTraversal.cpp
  05_TreeTraversal.exe
  06_EndTermPractical.cpp
  06_EndTermPractical.exe
  07_endTermPractical.cpp
  07_endTermPractical.exe
  > OUTLINE
TERMINAL
PS E:\lucky\my cpp\programs of C> cd "E:\lucky\my cpp\programs of C\Divya" & if ($?) { g++ 01_sorting.cpp -o 01_sorting } & if ($?) { \.\01_sorting }
Enter no. of elements u want to sort : 4
Enter 1 Element : 6
Enter 2 Element : 7
Enter 3 Element : 8
Enter 4 Element : 89
Please select any option Given Below for Sorting :
1. Bubble Sort
2. Selection Sort
3. Insertion Sort
4. Display Array.
5. Exit the Program.
Enter your Choice : 1
After Bubble sort Elements are : 6 7 8 89
1. Bubble Sort
2. Selection Sort
3. Insertion Sort
4. Display Array.
5. Exit the Program.
Enter your Choice : 3
After Insertion sort Elements are : 6 7 8 89
1. Bubble Sort
2. Selection Sort
3. Insertion Sort
4. Display Array.
5. Exit the Program.
Enter your Choice : 5
PS E:\lucky\my cpp\programs of C\Divya> []

```

Lab Assignment 9

01_binarySearchTree

```
// checking whether a tree is a BST?  
#include <stdio.h>  
#include <malloc.h>  
  
struct node  
{  
    int data;  
    struct node *left;  
    struct node *right;  
};  
  
struct node *createNode(int data)  
{  
    struct node *n; // creating a node pointer  
    n = (struct node *)malloc(sizeof(struct node)); // Allocating memory in the heap  
    n->data = data; // Setting the data  
    n->left = NULL; // Setting the left and right children to NULL  
    n->right = NULL; // Setting the left and right children to NULL  
    return n; // Finally returning the created node  
}  
  
void preOrder(struct node *root)  
{  
    if (root != NULL)  
    {
```

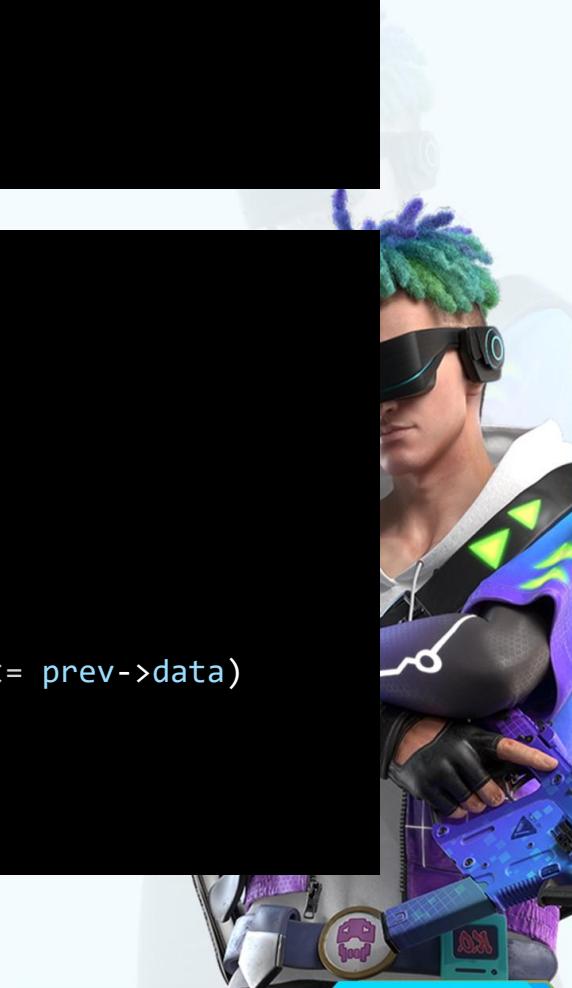


```
        printf("%d ", root->data);
        preOrder(root->left);
        preOrder(root->right);
    }
}
```

```
void postOrder(struct node *root)
{
    if (root != NULL)
    {
        postOrder(root->left);
        postOrder(root->right);
        printf("%d ", root->data);
    }
}
```

```
void inOrder(struct node *root)
{
    if (root != NULL)
    {
        inOrder(root->left);
        printf("%d ", root->data);
        inOrder(root->right);
    }
}
```

```
int isBST(struct node *root)
{
    static struct node *prev = NULL;
    if (root != NULL)
    {
        if (!isBST(root->left))
        {
            return 0;
        }
        if (prev != NULL && root->data <= prev->data)
        {
            return 0;
        }
        prev = root;
    }
}
```



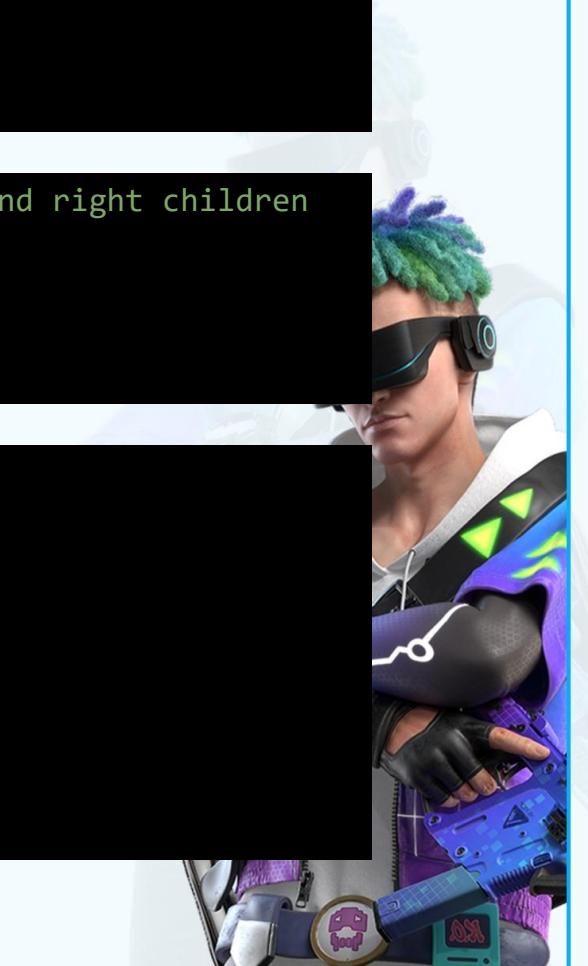
```
        return isBST(root->right);
    }
    else
    {
        return 1;
    }
}
```

```
int main()
{
```

```
    // Constructing the root node - Using Function (Recommended)
    struct node *p = createNode(5);
    struct node *p1 = createNode(3);
    struct node *p2 = createNode(6);
    struct node *p3 = createNode(1);
    struct node *p4 = createNode(4);
    // Finally The tree looks like this:
    //      5
    //      / \
    //      3   6
    //      / \
    //      1   4
```

```
    // Linking the root node with left and right children
    p->left = p1;
    p->right = p2;
    p1->left = p3;
    p1->right = p4;
```

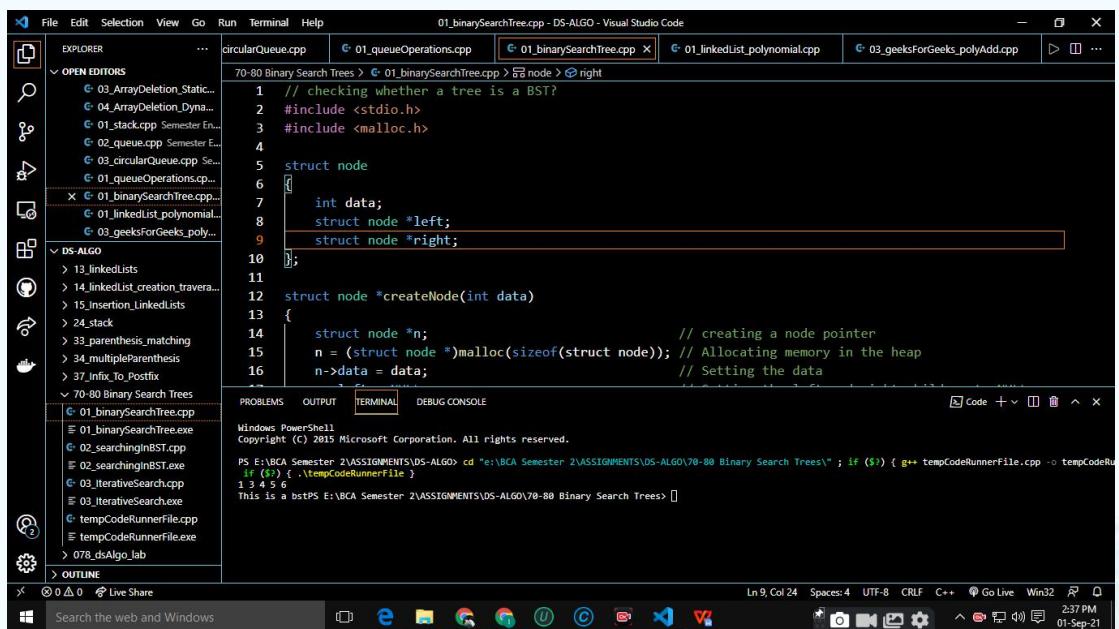
```
    // preOrder(p);
    // printf("\n");
    // postOrder(p);
    // printf("\n");
    inOrder(p);
    printf("\n");
    // printf("%d", isBST(p));
    if (isBST(p))
    {
```



```

        printf("This is a bst");
    }
else
{
    printf("This is not a bst");
}
return 0;
}

```



```

File Edit Selection View Go Run Terminal Help 01_binarySearchTree.cpp - DS-ALGO - Visual Studio Code
circularQueue.cpp 01_queueOperations.cpp 01_binarySearchTree.cpp x 01_linkedList_polynomial.cpp 03_geeksForGeeks_polyAdd.cpp
OPEN EDITORS
EXPLORER
OPENED FILES
01_ArrayDeletion_Static.cpp
01_ArrayDeletion_Dynamic.cpp
01_stack.cpp Semester End Test
02_queue.cpp Semester End Test
03_circularQueue.cpp Semester End Test
01_queueOperations.cpp
01_binarySearchTree.cpp
01_linkedList_polynomial.cpp
03_geeksForGeeks_polyAdd.cpp
DS-ALGO
13_linkedList
14_linkedList_creation_traversal
15_Insertion_LinkedLists
24_stack
33_parenthesis_matching
34_multipleParenthesis
37_Infix_To_Postfix
70-80 Binary Search Trees
01_binarySearchTree.cpp
01_binarySearchTree.exe
02_searchingInBST.cpp
02_searchingInBST.exe
03_IterativeSearch.cpp
03_IterativeSearch.exe
tempCodeRunnerFile.cpp
tempCodeRunnerFile.exe
078_dsAlgo_lab
OUTLINE
PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE
Windows PowerShell
Copyright (C) 2015 Microsoft Corporation. All rights reserved.
PS E:\BCA Semester 2\ASSIGNMENTS\DS-ALGO> cd "e:\BCA Semester 2\ASSIGNMENTS\DS-ALGO\70-80 Binary Search Trees" ; If ($?) { g++ tempCodeRunnerFile.cpp -o tempCodeRunnerFile
1 2 3 4 5 6
This is a bstPS E:\BCA Semester 2\ASSIGNMENTS\DS-ALGO\70-80 Binary Search Trees>
Ln 9, Col 24 Spaces:4 UTF-8 CRLF C++ Go Live Win32
2:37 PM 01-Sep-21

```

02_searchingInBST

```

#include <stdio.h>
#include <malloc.h>

struct node
{
    int data;
    struct node *left;
    struct node *right;
};

struct node *createNode(int data)

```

```
{  
    struct node *n; // creating a node pointer  
    n = (struct node *)malloc(sizeof(struct node)); // Allocating memory in the heap  
    n->data = data; // Setting the data  
    n->left = NULL; // Setting the left and right children to NULL  
    n->right = NULL; // Setting the left and right children to NULL  
    return n; // Finally returning the created node  
}
```

```
void preOrder(struct node *root)  
{  
    if (root != NULL)  
    {  
        printf("%d ", root->data);  
        preOrder(root->left);  
        preOrder(root->right);  
    }  
}
```

```
void postOrder(struct node *root)  
{  
    if (root != NULL)  
    {  
        postOrder(root->left);  
        postOrder(root->right);  
        printf("%d ", root->data);  
    }  
}
```

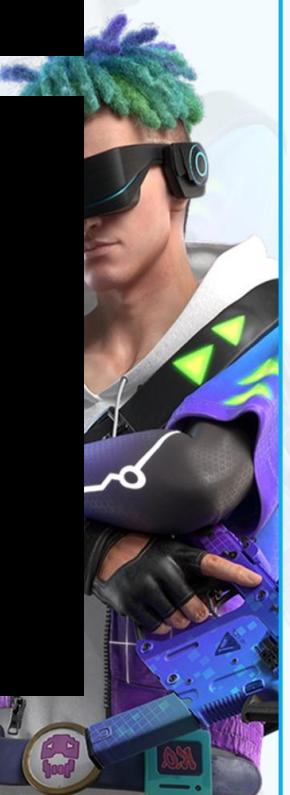
```
void inOrder(struct node *root)  
{  
    if (root != NULL)  
    {  
        inOrder(root->left);  
    }
```



```
        printf("%d ", root->data);
        inOrder(root->right);
    }
}
```

```
int isBST(struct node *root)
{
    static struct node *prev = NULL;
    if (root != NULL)
    {
        if (!isBST(root->left))
        {
            return 0;
        }
        if (prev != NULL && root->data <= prev->data)
        {
            return 0;
        }
        prev = root;
        return isBST(root->right);
    }
    else
    {
        return 1;
    }
}
```

```
struct node *search(struct node *root, int key)
{
    if (root == NULL)
    {
        return NULL;
    }
    if (key == root->data)
    {
        return root;
    }
    else if (key < root->data)
    {
        return search(root->left, key);
    }
}
```



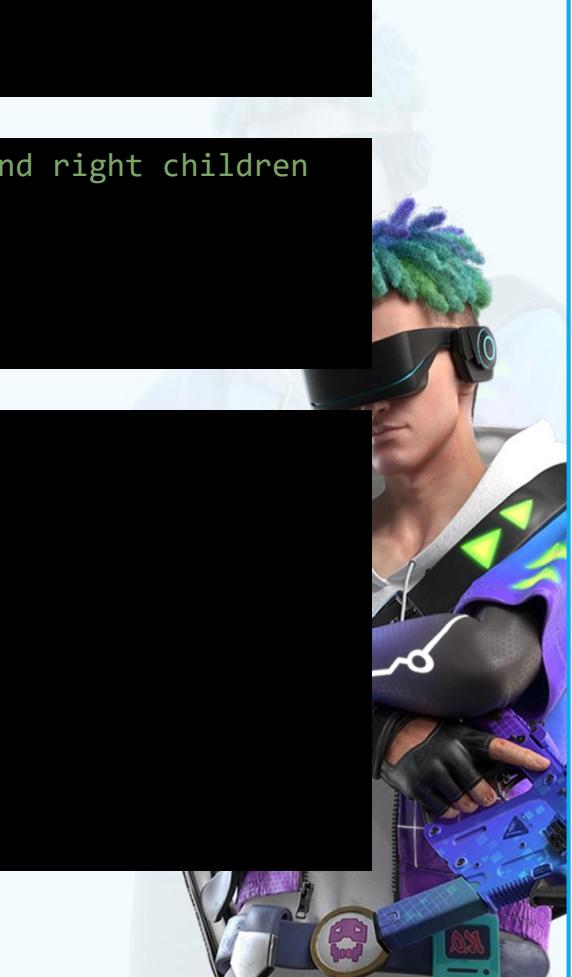
```
        }
    else
    {
        return search(root->right, key);
    }
}
```

```
int main()
{
```

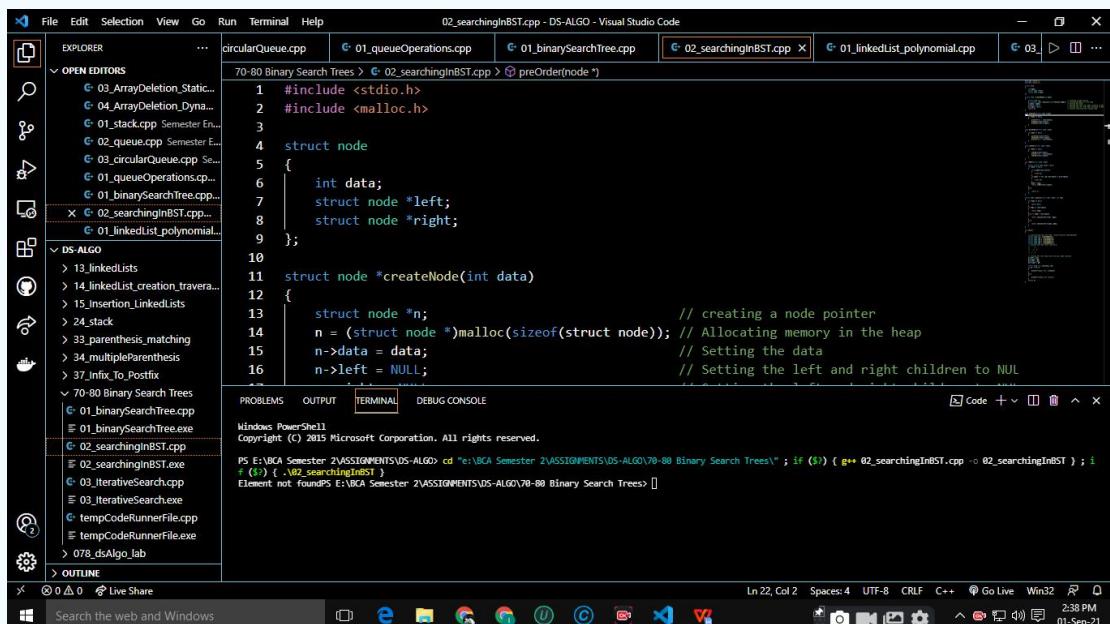
```
    // Constructing the root node - Using Function (Recommended)
    struct node *p = createNode(5);
    struct node *p1 = createNode(3);
    struct node *p2 = createNode(6);
    struct node *p3 = createNode(1);
    struct node *p4 = createNode(4);
    // Finally The tree looks like this:
    //      5
    //      / \
    //      3   6
    //      / \
    //      1   4
```

```
    // Linking the root node with left and right children
    p->left = p1;
    p->right = p2;
    p1->left = p3;
    p1->right = p4;
```

```
    struct node *n = search(p, 10);
    if (n != NULL)
    {
        printf("Found: %d", n->data);
    }
    else
    {
        printf("Element not found");
    }
    return 0;
```



{}



```

File Edit Selection View Go Run Terminal Help 02_searchingInBST.cpp - DS-ALGO - Visual Studio Code
circularQueue.cpp 01_queueOperations.cpp 01_binarySearchTree.cpp 02_searchingInBST.cpp X 01_linkedList_polynomial.cpp 03...
OPEN EDITORS
03_ArrayDeletion_Static...
04_ArrayDeletion_Dyna...
01_stack.cpp Semester E...
02_queue.cpp Semester E...
03_circularQueue.cpp Semester E...
01_queueOperations.cpp Semester E...
01_binarySearchTree.cpp Semester E...
02_searchingInBST.cpp X Semester E...
01_linkedList_polynomial...
DS-ALGO
13.linkedLists
14.linkedList_creation_traversal
15.Insertion_LinkedLists
24.stack
33_parenthesis_matching
34_multipleParenthesis
37_infix_to_Postfix
70-80 Binary Search Trees
01_binarySearchTree.cpp
01_binarySearchTree.exe
02_searchingInBST.cpp
02_searchingInBST.exe
03_IterativeSearch.cpp
03_IterativeSearch.exe
tempCodeRunner.cpp
tempCodeRunnerFile.exe
078.dsAlgo_lab
OUTLINE
PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE
Windows PowerShell
Copyright (C) 2015 Microsoft Corporation. All rights reserved.
PS E:\VBCA Semester 2\ASSIGNMENTS\DS-ALGO> cd "E:\VBCA Semester 2\ASSIGNMENTS\DS-ALGO\70-80 Binary Search Trees">; If ($_) { g++ 02_searchingInBST.cpp -o 02_searchingInBST } ; i
Element not foundPS E:\VBCA Semester 2\ASSIGNMENTS\DS-ALGO\70-80 Binary Search Trees> []
Ln 22, Col 2 Spaces: 4 UTF-8 CRLF C++ Go Live Win32 2:38 PM 01-Sep-21

```

05_TreeTraversal

```

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

struct node
{
    int data;
    struct node *left;
    struct node *right;
};

```

```

struct node *temp;
int num;

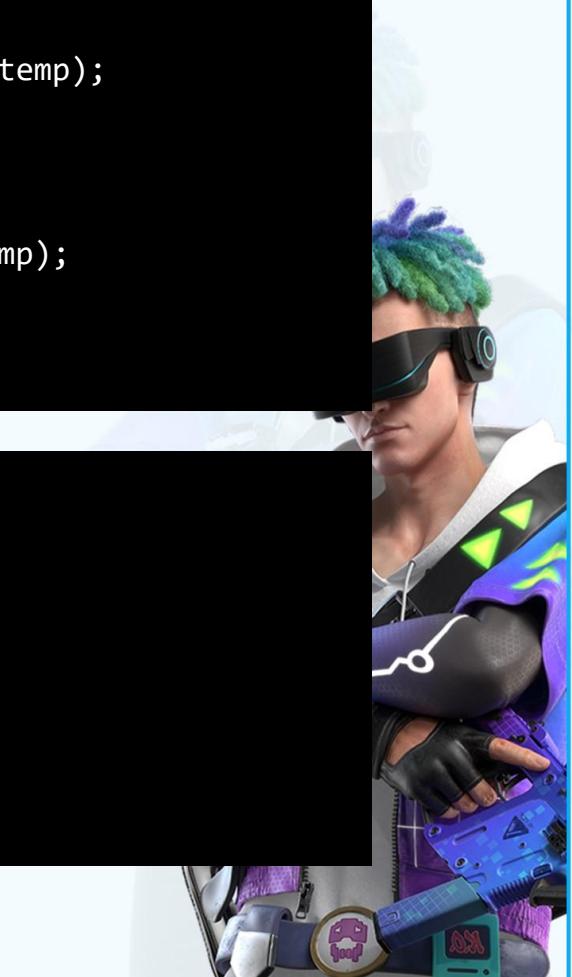
```

```
struct node *create(int num)
{
    temp = (struct node *)malloc(sizeof(struct node));
    if (temp == NULL)
        printf("not enough memory\n");
    else
    {
        temp->data = num;
        temp->left = NULL;
        temp->right = NULL;
    }
    return (temp);
}
```

```
struct node* insert(struct node *root, struct node *temp1)
{
    if (root ==NULL) return temp;

    if (temp1->data > root->data)
    {
        root->right=insert(root->right, temp);
    }
    else
    {
        root->left=insert(root->left, temp);
    }
    return root;
}
```

```
void preorder(struct node *root)
{
    if (root != NULL)
    {
        printf("%d\n", root->data);
        preorder(root->left);
        preorder(root->right);
    }
}
```



```
void inorder(struct node *root)
{
    if (root != NULL)
    {
        inorder(root->left);
        printf("%d\n", root->data);
        inorder(root->right);
    }
}
```

```
void postorder(struct node *root)
{
    if (root != NULL)
    {
        postorder(root->left);
        postorder(root->right);
        printf("%d\n", root->data);
    }
}
```

```
struct node *minValueNode(struct node *root)
{
    struct node *current = root;
```

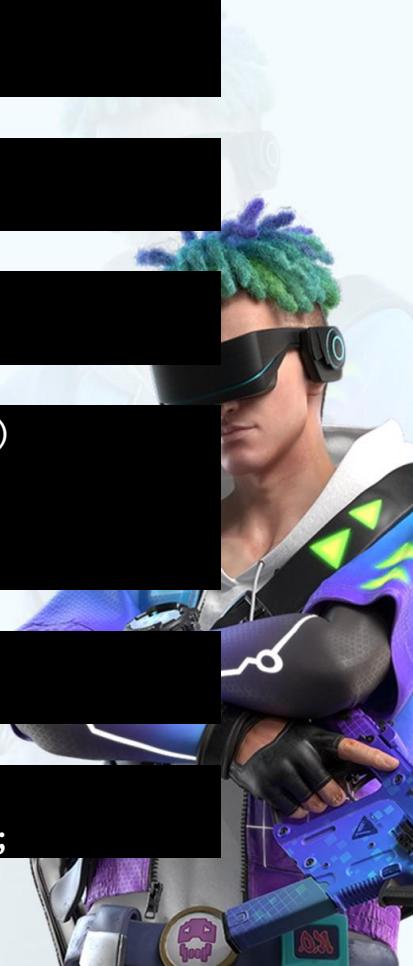
```
    while (current && current->left != NULL)
        current = current->left;

    return current;
}
```

```
struct node *deleteNode(struct node *root, int a)
{
    if (root == NULL)
        return root;

    if (a < root->data)
        root->left = deleteNode(root->left, a);

    else if (a > root->data)
        root->right = deleteNode(root->right, a);
```



```

else
{
    // node with only one child or no child
    if (root->left == NULL)
    {
        temp = root->right;
        free(root);
        return temp;
    }
    else if (root->right == NULL)
    {
        temp = root->left;
        free(root);
        return temp;
    }
}

// node with two children:
// Get the inorder successor
// (smallest in the right subtree)
temp = minValueNode(root->right);

// Copy the inorder
// successor's content to this node
root->data = temp->data;

// Delete the inorder successor
root->right = deleteNode(root->right, temp->data);
}

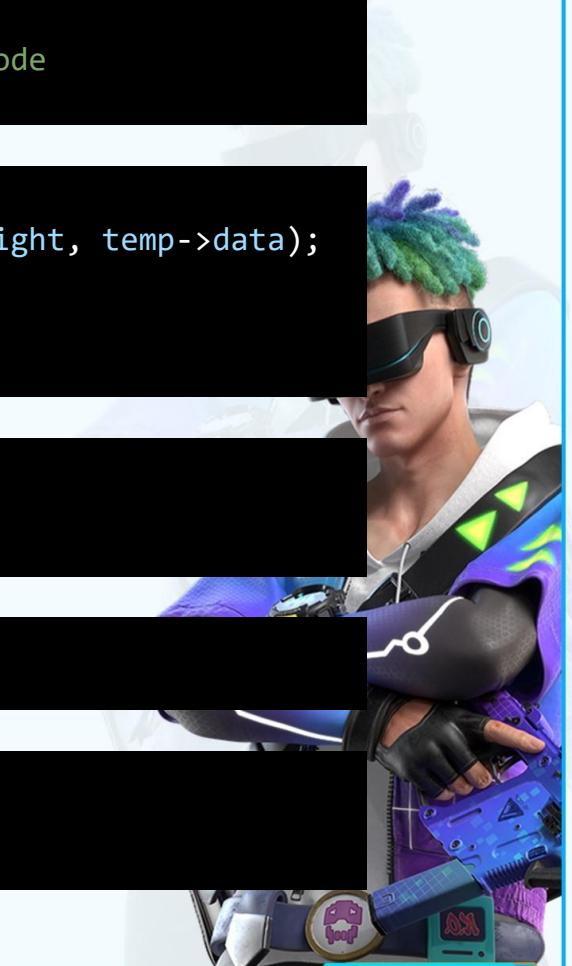
return root;
}

int totalNodes(struct node *root)
{
    int count = 0;

    if (root == NULL)
        return 0;

    else
    {
        count+=1;
    }
}

```



```
        count += totalNodes(root->left);
        count += totalNodes(root->right);
        return count;
    }
}
```

```
void deleteTree(struct node *root)
{
    if (root == NULL)
        return;

    /* first delete both subtrees */
    deleteTree(root->left);
    deleteTree(root->right);

    /* then delete the node */
    printf("\n Deleting node: %d", root->data);
    free(root);
}
```

```
int main()
{
    int ch, val;
    struct node *root = NULL, *temp;

    do
    {
        printf("\n **MAIN MENU* \n");
        printf("1. create/Insert element\n");
        printf("2. Preorder Traversal\n");
        printf("3. Inorder Traversal\n");
        printf("4. Postorder Traversal\n");
        printf("5. Delete an element\n");
        printf("6. Count the total number of nodes\n");
        printf("7. Delete the tree\n");
        printf("8. Exit\n");
        printf("Enter your option :\n ");
}
```

```

        scanf("%d", &ch);
        switch (ch)
        {
            case 1:
                printf("Enter the element\n");
                scanf("%d", &num);
                temp = create(num);
                root=insert(root, temp);
                break;

            case 2:
                printf("\n The elements of the tree are : \n");
                preorder(root);
                break;

            case 3:
                printf("\n The elements of the tree are : \n");
                inorder(root);
                break;

            case 4:
                printf("\n The elements of the tree are : \n");
                postorder(root);
                break;

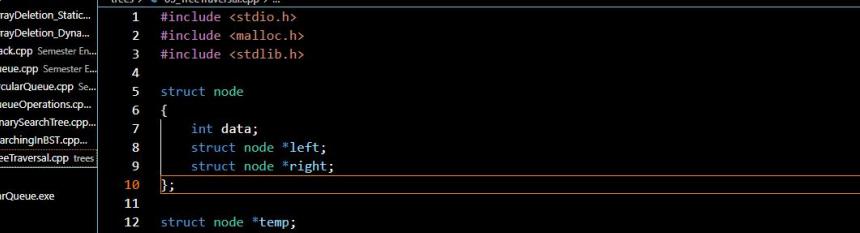
            case 5:
                printf("\n Enter the element to be deleted : ");
                scanf("%d", &val);
                root = deleteNode(root, val);
                break;

            case 6:
                printf("\n Total no. of nodes = %d", totalNodes(
root));
                break;

            case 7:
                deleteTree(root);
                break;
        }
    } while (ch != 8);

```

```
    return 0;  
}
```



File Edn Selection View Go Run Terminal Help 05_TreeTraversal.cpp - DS-ALGO - Visual Studio Code

EXPLORER OPEN EDITORS circularQueue.cpp 01_queueOperations.cpp 01_binarySearchTree.cpp 02_searchingInBST.cpp 05_TreeTraversal.cpp 01_linkedLists.cpp

03.circularQueue.exe 04.ArrayDeletion_Static... 04.ArrayDeletion_Dyna... 01.stack.cpp Semester En... 02.queue.cpp Semester E... 03.circularQueue.cpp Semester E... 01.queueOperations.cpp 01.binarySearchTree.cpp 02.searchingInBST.cpp 05.TreeTraversal.cpp trees

05.TreeTraversal.cpp trees > 05.TreeTraversal.cpp > ...

```
1 #include <stdio.h>
2 #include <malloc.h>
3 #include <stdlib.h>
4
5 struct node
6 {
7     int data;
8     struct node *left;
9     struct node *right;
10 };
11
12 struct node *temp;
13 int num;
14
15 struct node *create(int num)
16 {
17 }
```

PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE

3 3

www.HWUP

1. create/Insert element
2. Preorder Traversal
3. Inorder Traversal
4. Postorder Traversal
5. Delete element
6. Count the total number of nodes
7. Delete the tree
8. Exit

Enter your option :

1

Ln 10, Col 3 Spaces:4 UTF-8 CRLF C++ Go Live Win32 2:40 PM 01-Sep-21

Search the web and Windows

Rest of the lab practicals are at : [here](#)

