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

Graphic Era
HILL UNIVERSITY
Established by an Act of the State Legislature of Uttarakhand (Adhiniyam Sankhya 12 of 2011)

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

Data Structures And File Organizations

TBC 201

INDEX

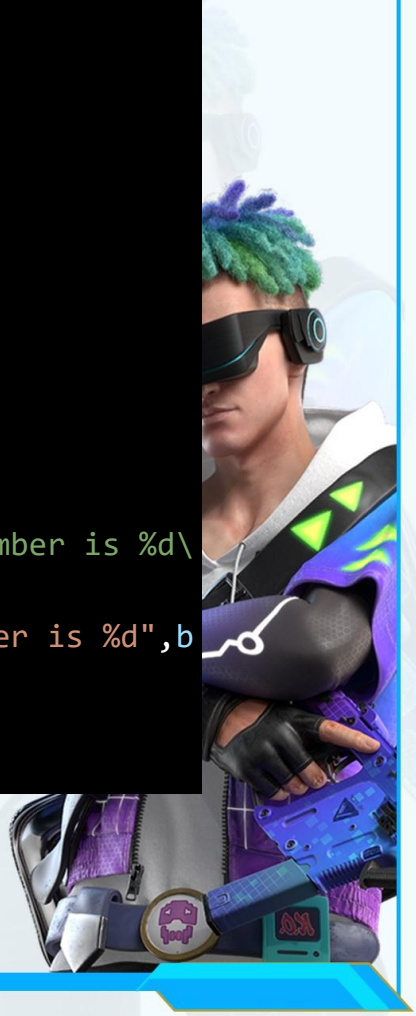
Sr. No.	Assign. Date	Subm. Date	Title	Teacher's Remark
1.		23 APRIL 2021	Lab Assignment 1	
2.		30 APRIL 2021	Lab Assignment 2	
3.		13 MAY 2021	Lab Assignment 3	
4.		13 MAY 2021	Lab Assignment 4	
5.		21 MAY 2021	Lab Assignment 5	
6.		26 MAY 2021	Lab Assignment 6	
7.		27 MAY 2021	Lab Assignment 7	
8.	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

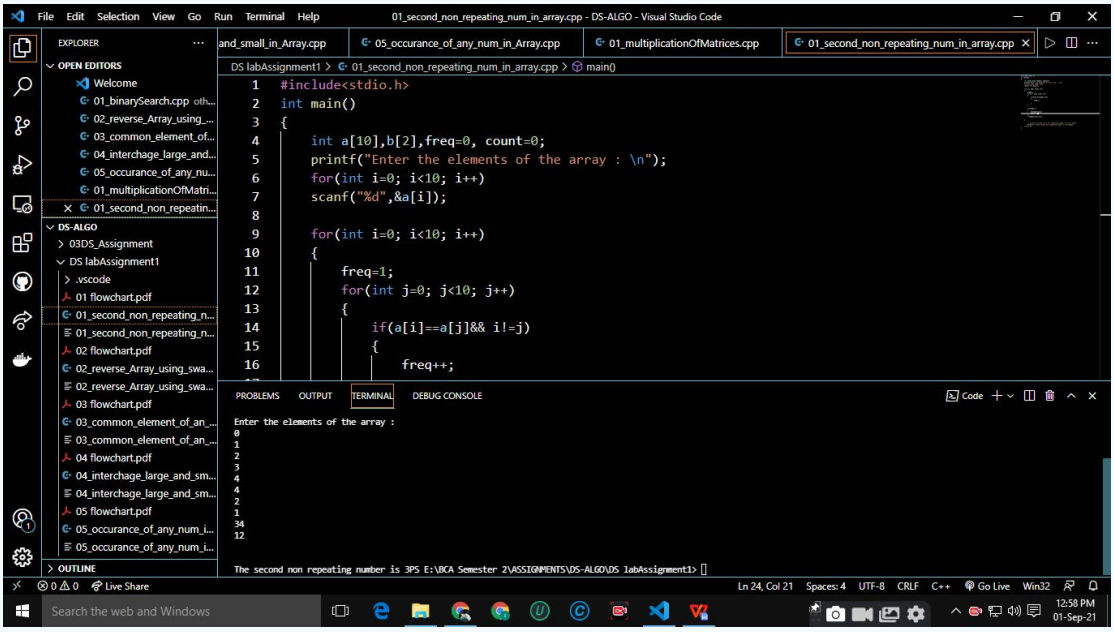
9.		**27 August 2021**	**Lab Assignment 9**	

Lab Assignment 1

01_second_non_repeating_num_in_array

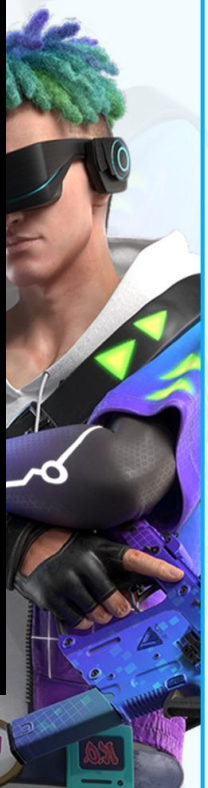
```c
#include<stdio.h>
int main()
{
    int a[10],b[2],freq=0, count=0;
    printf("Enter the elements of the array : \n");
    for(int i=0; i<10; i++)
    scanf("%d",&a[i]);

    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;
}
```



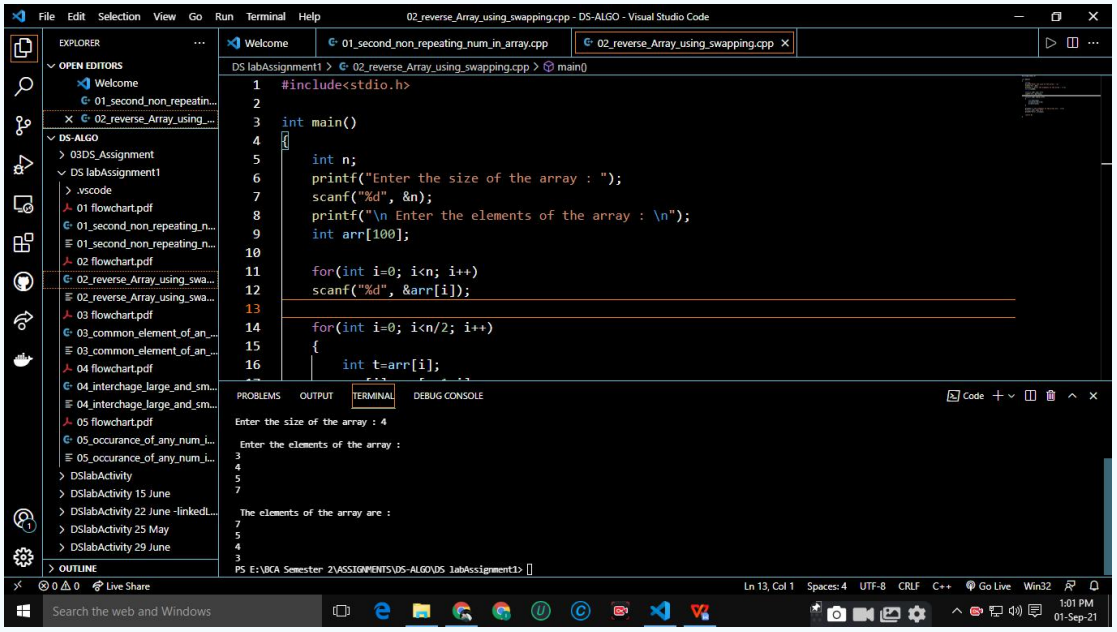


02_reverse_Array_using_swapping

```c
#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]);
```

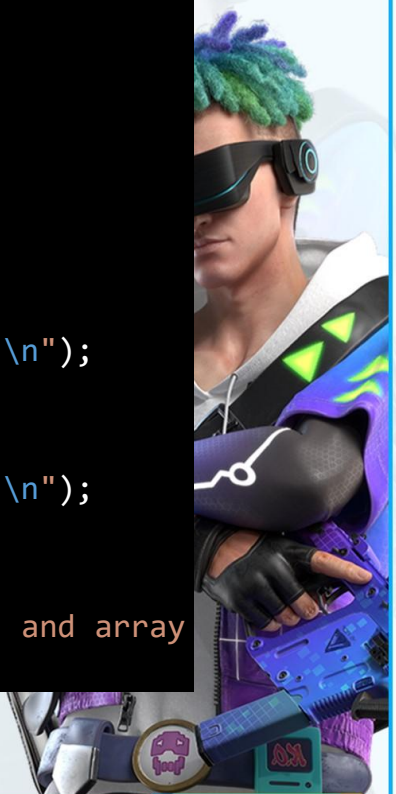


```
        return 0;
}
```

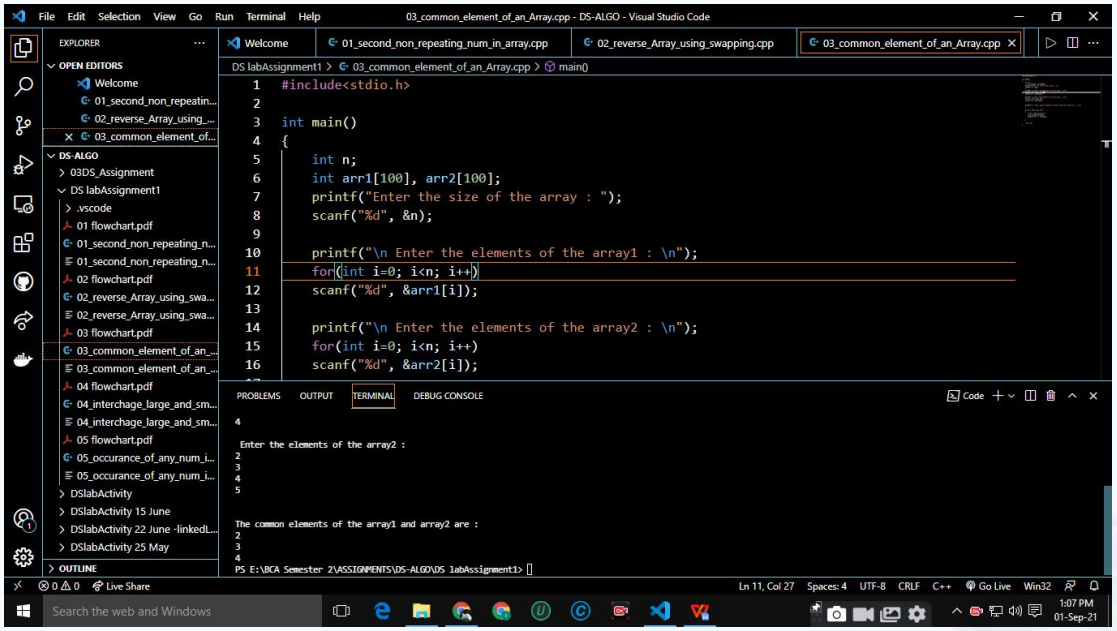


03_common_element_of_an_Array

```c
#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");
```

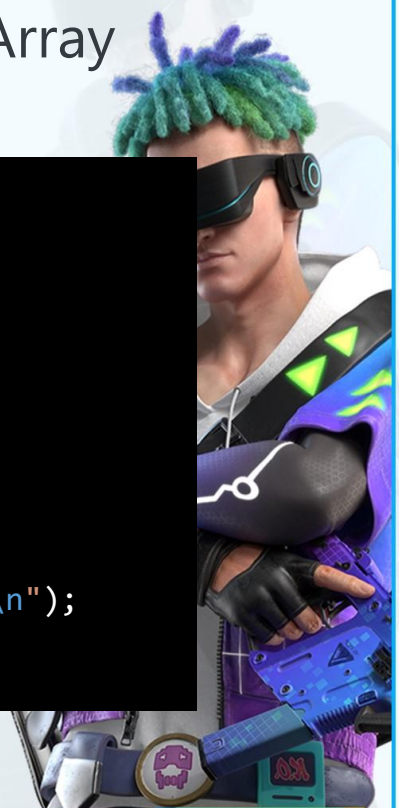


```
    for(int i=0; i<n; i++)
    {
        for(int j=0; j<n; j++)
        if(arr1[i]==arr2[j])
        printf("%d\n", arr1[i]);
    }
    return 0;
}
```



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]);
```



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

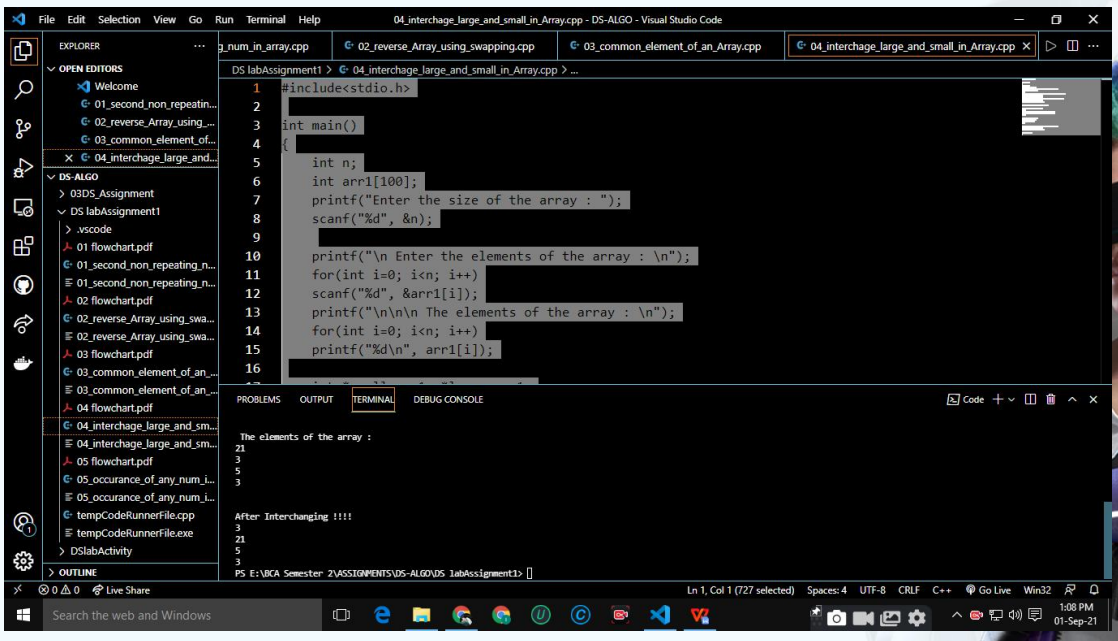
```c
    int *small=arr1, *large=arr1;
```

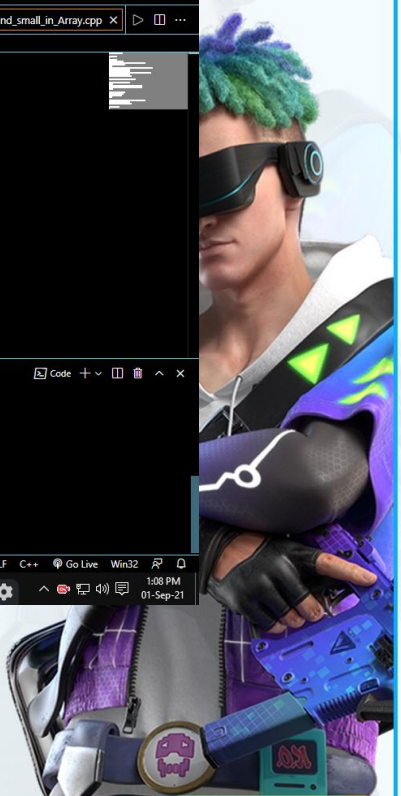
```c
    for(int i=0; i<n; i++)
    {
        if(arr1[i]<*small)small=&arr1[i];
        if(arr1[i]>*large)large=&arr1[i];
    }
```

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

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

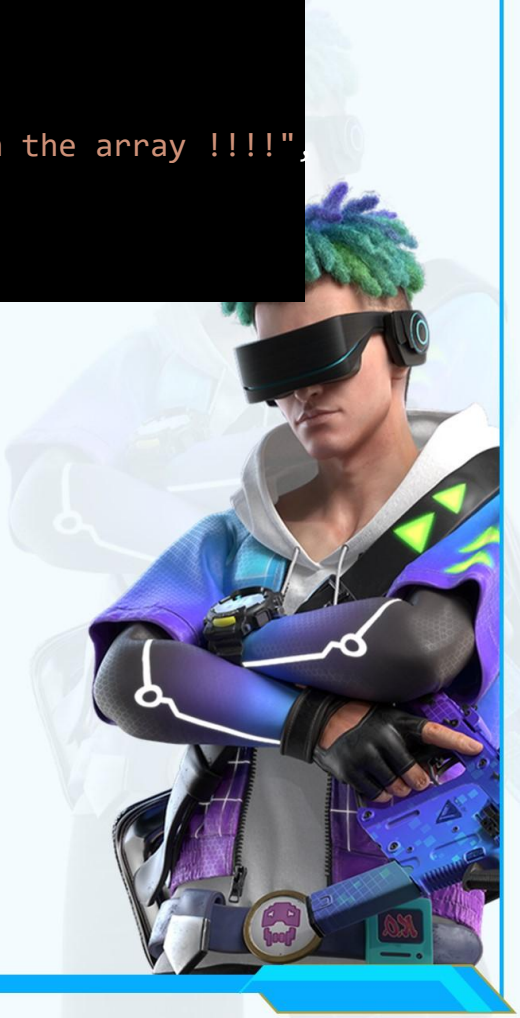
```c
    return 0;
}
```

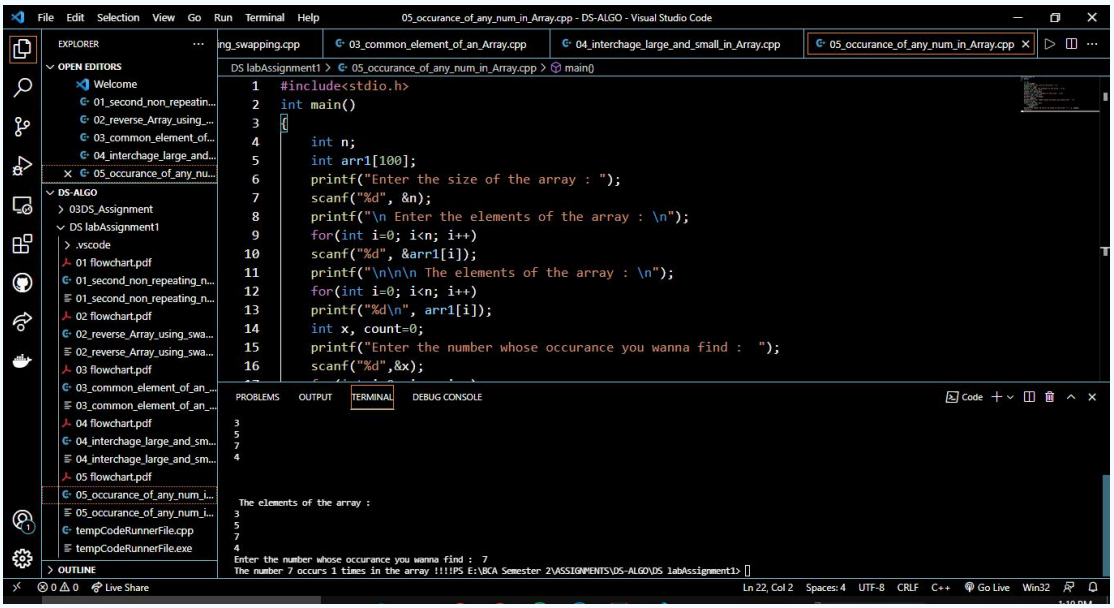




05_occurance_of_any_num_in_Array

```c
#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 occurance 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;
}
```




File Edit Selection View Go Run Terminal Help
05_occurance_of_any_num_in_Array.cpp - DS-ALGO - Visual Studio Code
EXPLORER
OPEN EDITORS
Welcome
01_second_non_repeatin...
02_reverse_Array_using_...
03_common_element_of...
04_interchage_large_and...
05_occurance_of_any_nu...
DS-ALGO
03DS_Assignment
DS labAssignment1
.vscode
01 flowchart.pdf
01_second_non_repeating_n...
01_second_non_repeating_n...
02 flowchart.pdf
02_reverse_Array_using_swa...
02_reverse_Array_using_swa...
03 flowchart.pdf
03_common_element_of_an_...
03_common_element_of_an_...
04 flowchart.pdf
04_interchage_large_and_sm...
04_interchage_large_and_sm...
05 flowchart.pdf
05_occurance_of_any_num_i...
05_occurance_of_any_num_i...
tempCodeRunnerFile.cpp
tempCodeRunnerFile.exe
OUTLINE
ng_swapping.cpp
03_common_element_of_an_Array.cpp
04_interchage_large_and_small_in_Array.cpp
05_occurance_of_any_num_in_Array.cpp
DS labAssignment1 > 05_occurance_of_any_num_in_Array.cpp > main()
#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 occurance you wanna find : ");
scanf("%d",&x);
PROBLEMS
OUTPUT
TERMINAL
DEBUG CONSOLE
Code
3
5
7
4
The elements of the array :
3
5
7
4
Enter the number whose occurance you wanna find : 7
The number 7 occurs 1 times in the array !!!PS E:\BCA Semester 2\ASSIGNMENTS\DS-ALGO\DS labAssignment1>
Live Share
Ln 22, Col 2
Spaces: 4
UTF-8
CRLF
C++
Go Live
Win32

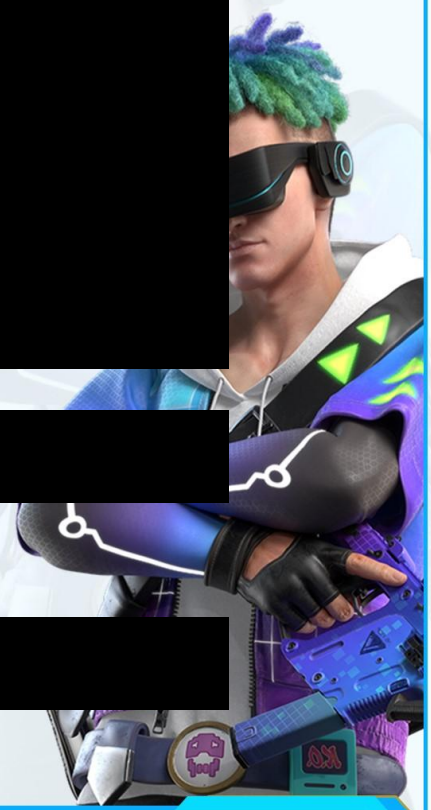
Lab Assignment 2

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);
```

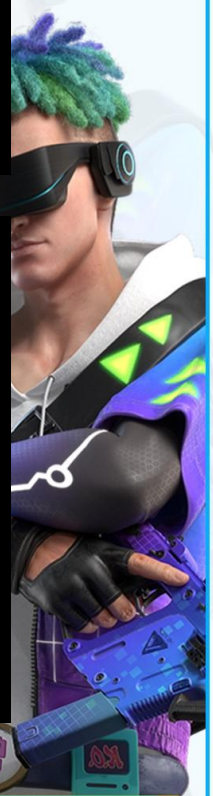
```
    printf("Enter the number of elements in the row & column
of matrix 2 : \n");
    scanf("%d",&r2);
    scanf("%d", &c2);
```

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

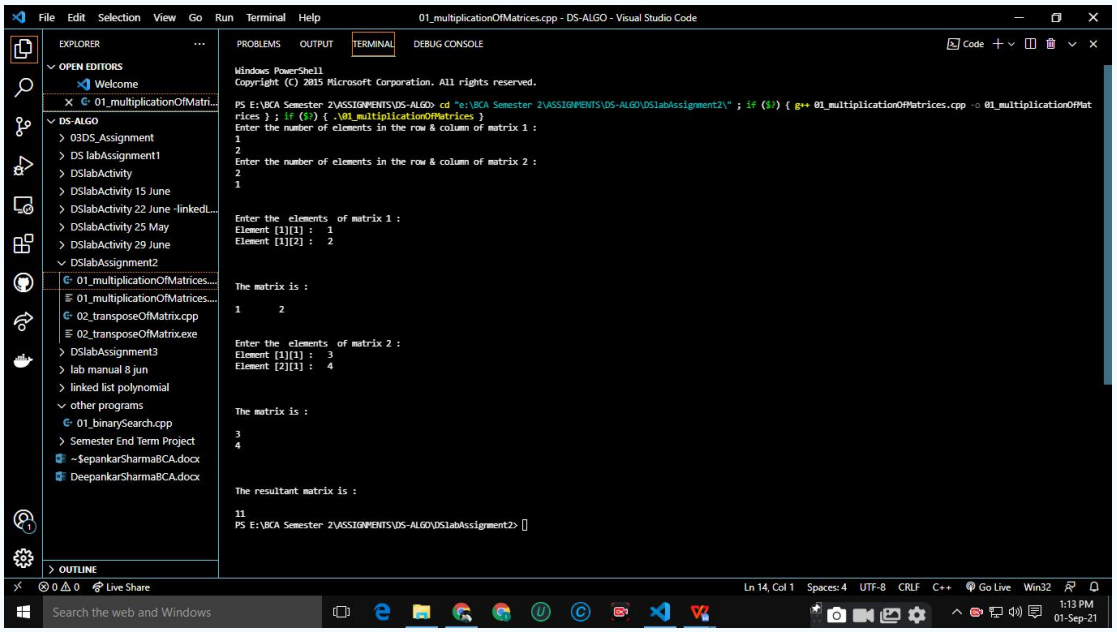
```
    printf("\n\nEnter the  elements  of matrix 1 : \n");
    mat_elem (a1  ,r1,c1);
    printf("\n\n\nThe matrix is : \n\n");
    show_elem(a1,r1,c1);
```

```
    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");
```

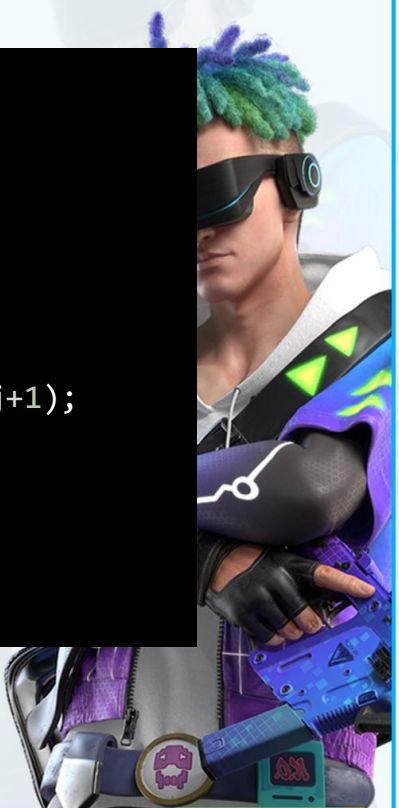


```
    show_elem(a3,r1,c2);
    }
```



02_transposeOfMatrix

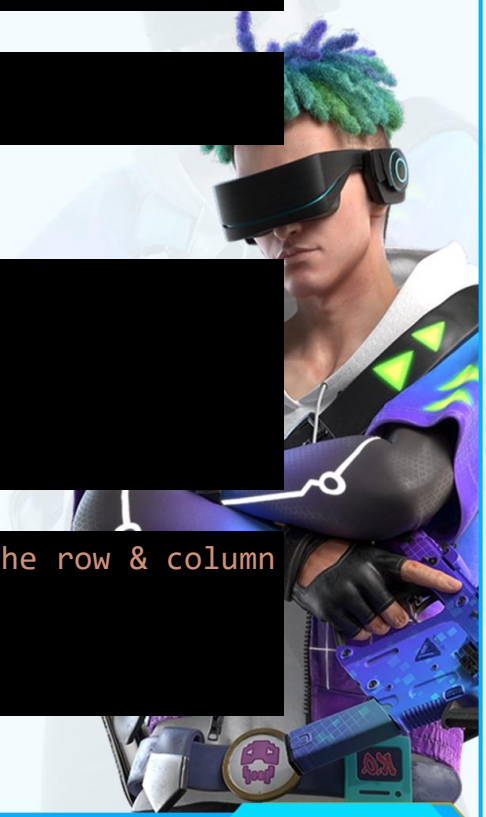
```
#include<stdio.h>
void mat_elem(int a[10][10], int r, int c)
{
    for(int i =0; i<r; i++)
    {
        for(int j=0; j<c; j++)
        {
            printf("Element [%d][%d] :   ", i+1, j+1);
            scanf("%d", &a[i][j] );

        }
    }
}
```



```c
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");

    }
}
```

```c
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");

    }
}
```

```c
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;
    // }
```

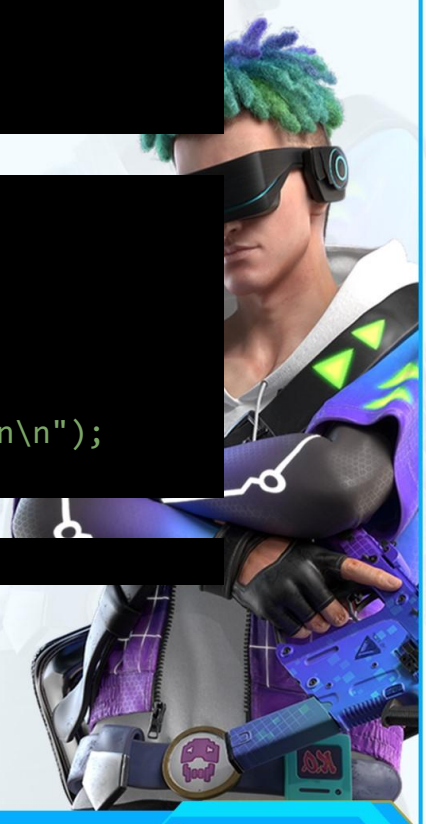
```
    printf("\n\nEnter the  elements  of matrix 1 : \n");
    mat_elem (a1  ,r1,c1);
    printf("\n\n\nThe matrix is : \n\n");
    show_elem(a1,r1,c1);
    tps_mat(a1,r1,c1);
```

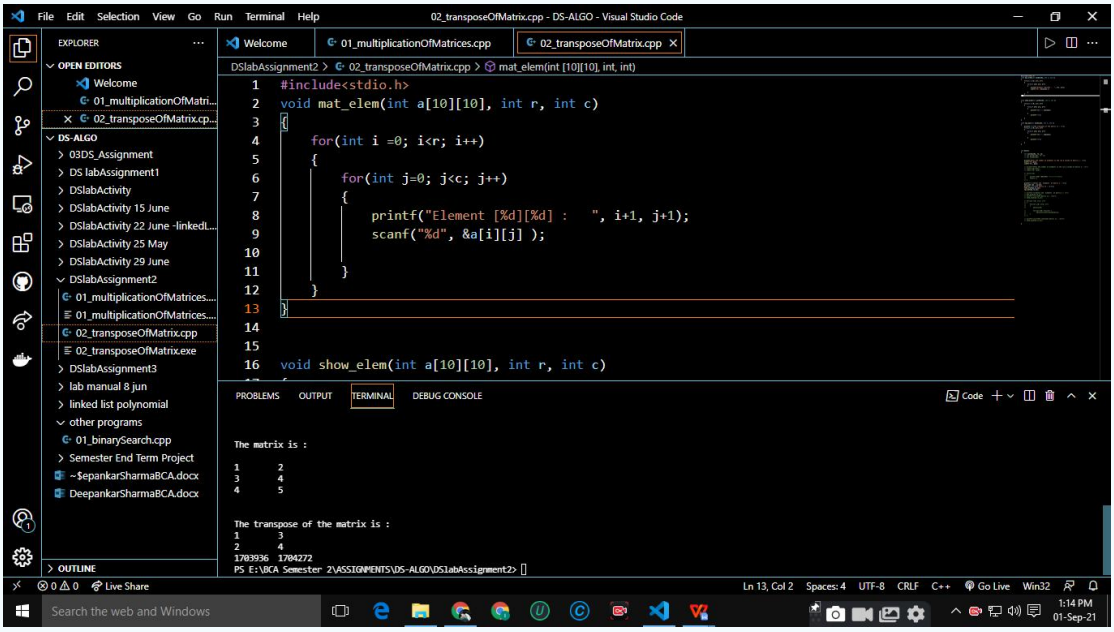
```
    // printf("\n\nEnter the  elements  of matrix 2 : \n");
    // mat_elem(a2,r2,c2);
    // printf("\n\n\nThe matrix is : \n\n");
    // show_elem(a2,r2,c2);
```

```
    // 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);
```

```
}
```





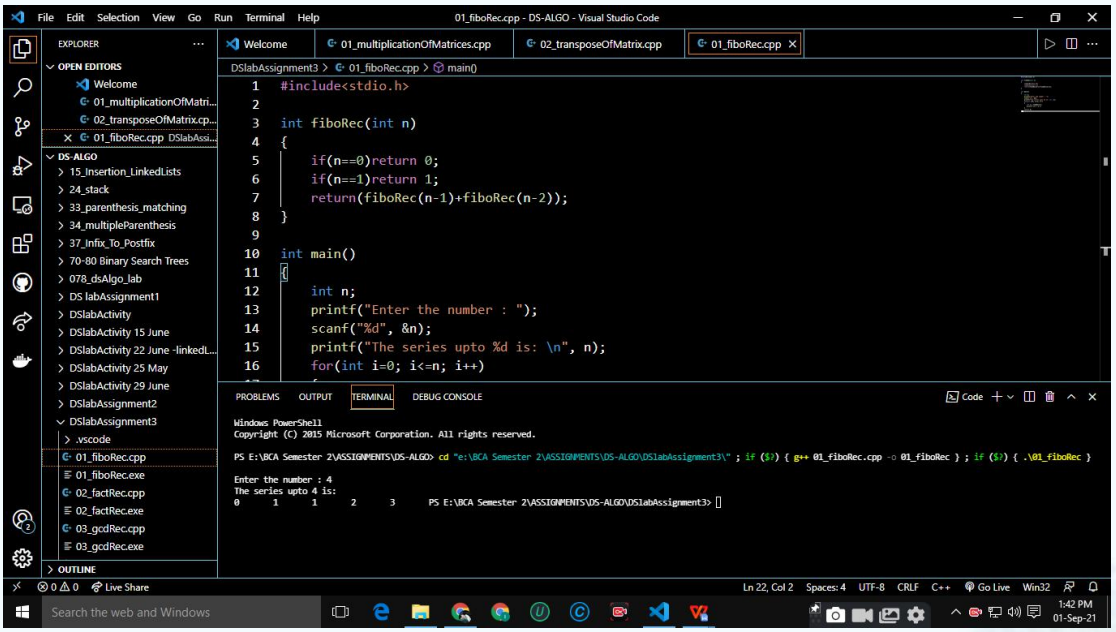
Lab Assignment 3

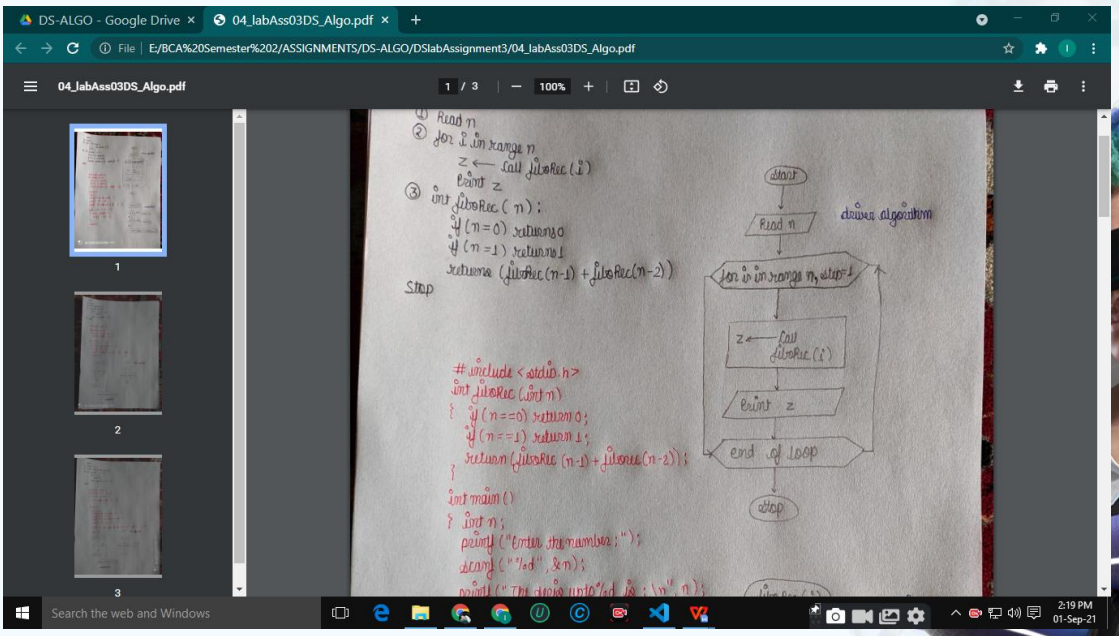
01_fibonacci_using_Recursion

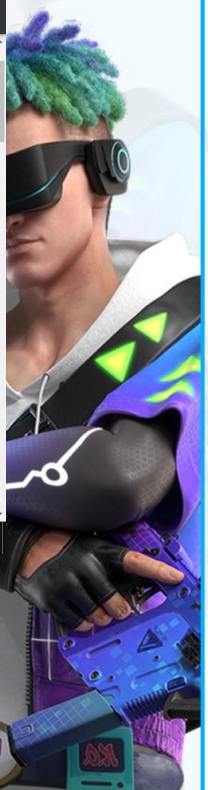
```c
#include<stdio.h>

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

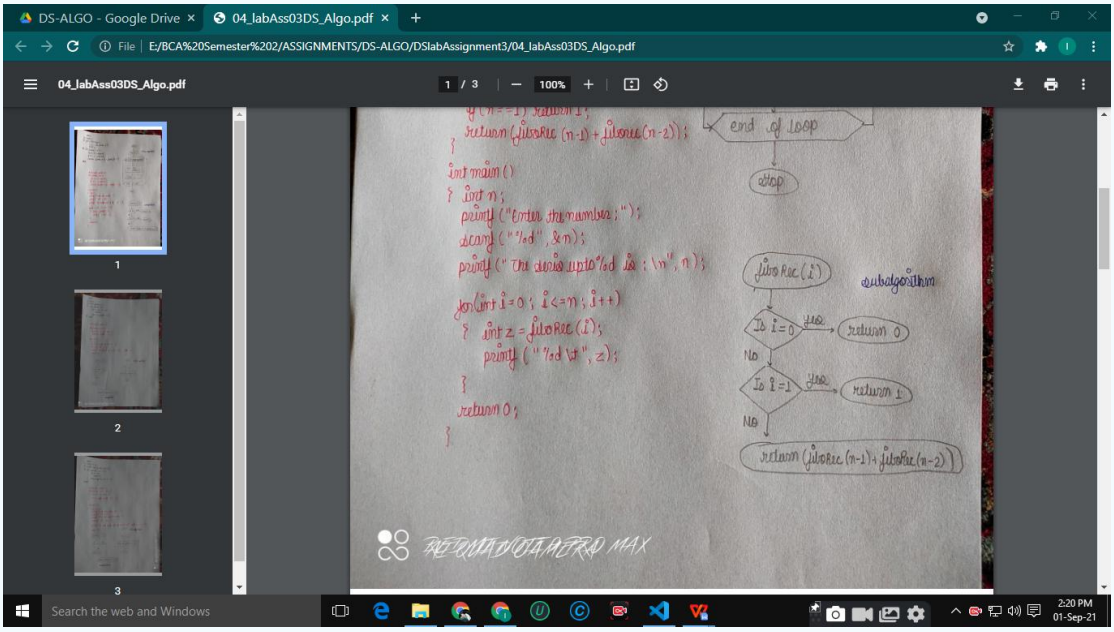
```c
int main()
{
    int n;
    printf("Enter the number : ");
    scanf("%d", &n);
    printf("The series upto %d is: \n", n);
```

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





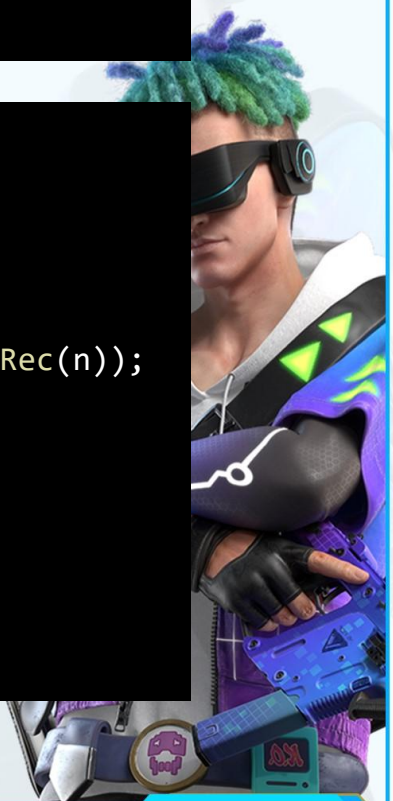


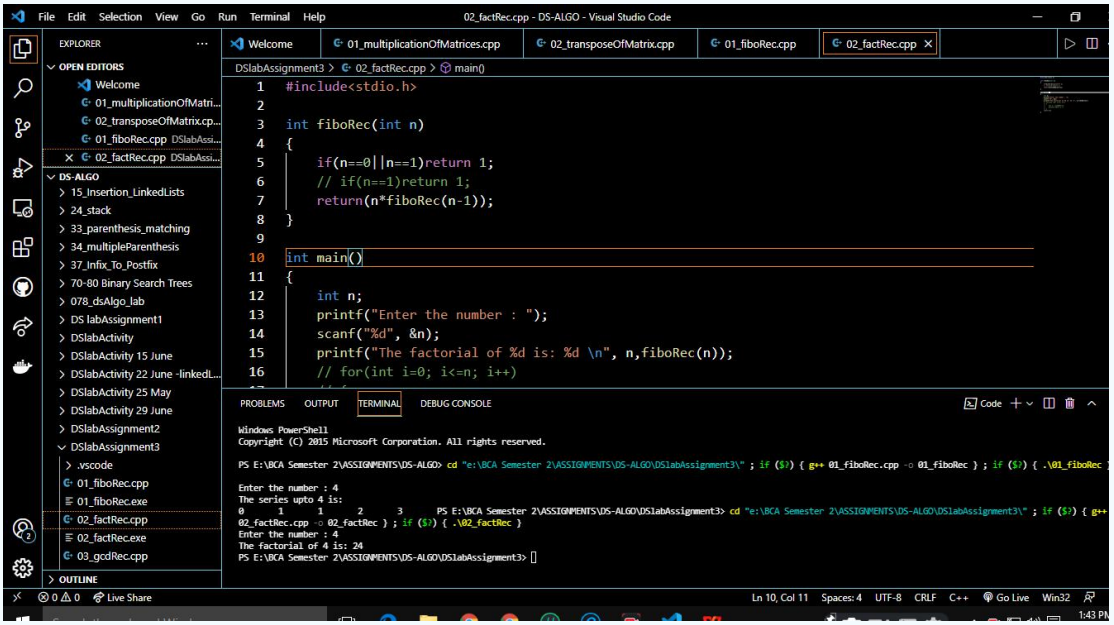


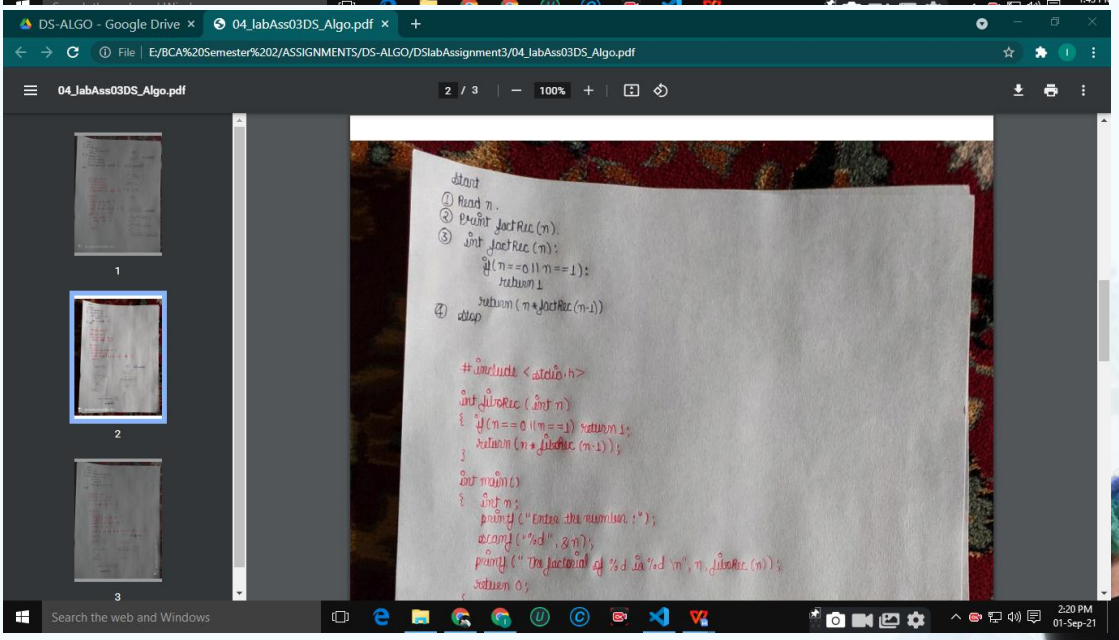
02_factRec

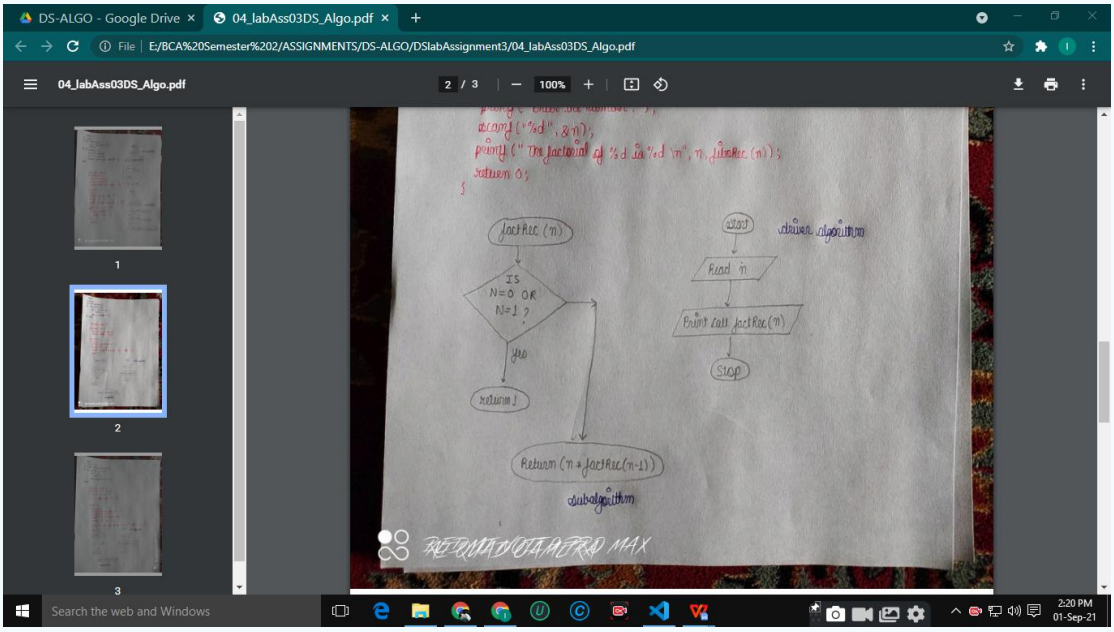
```c
#include<stdio.h>

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

```c
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 Edit 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
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
DSlabActivity
DSlabActivity 15 June
DSlabActivity 22 June -linkedL...
DSlabActivity 25 May
DSlabActivity 29 June
DSlabAssignment2
DSlabAssignment3
.vscode
01_fiboRec.cpp
01_fiboRec.exe
02_factRec.cpp
02_factRec.exe
03_gcdRec.cpp
OUTLINE
DSlabAssignment3 > 02_factRec.cpp > main()
#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++)
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 ($?) { g++ 01_fiboRec.cpp -o 01_fiboRec } ; if ($?) { .\01_fiboRec
Enter the number : 4
The series 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>
Live Share
Ln 10, Col 11 Spaces: 4 UTF-8 CRLF C++ Go Live Win32

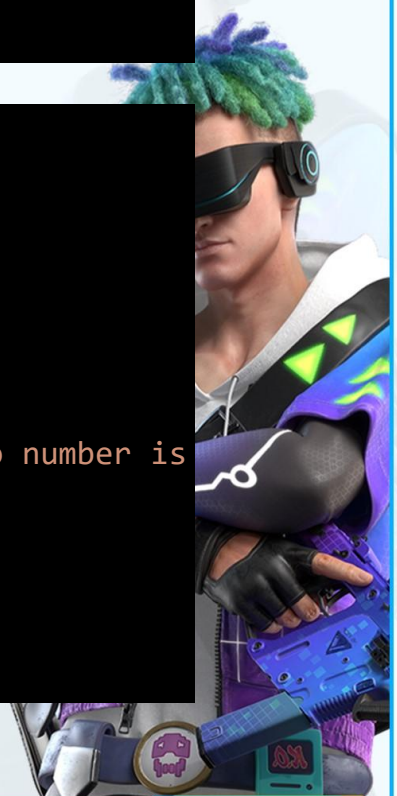

DS-ALGO - Google Drive
04_labAss03DS_Algo.pdf
File E:/BCA%20Semester%202/ASSIGNMENTS/DS-ALGO/DSlabAssignment3/04_labAss03DS_Algo.pdf
04_labAss03DS_Algo.pdf
2 / 3
100%
1
2
3
start
Read n.
Print factRec(n).
int factRec(n):
if(n==0||n==1):
return 1
return (n*factRec(n-1))
stop
#include <stdio.h>
int fiboRec(int n)
{ if(n==0||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));
return 0;
Search the web and Windows
2:20 PM
01-Sep-21



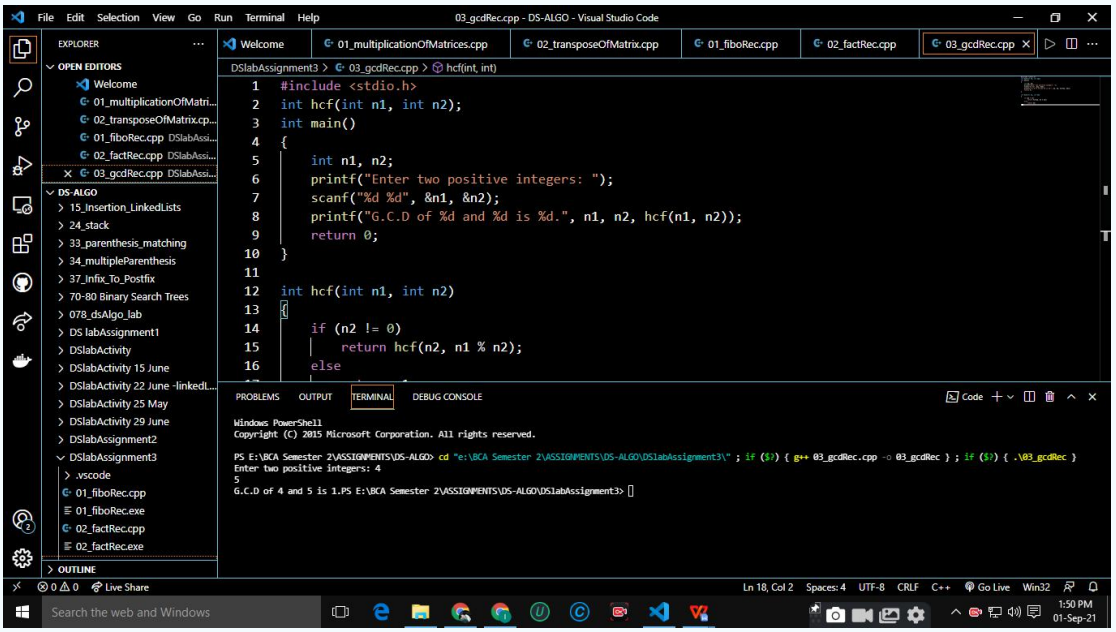
03_gcdRec

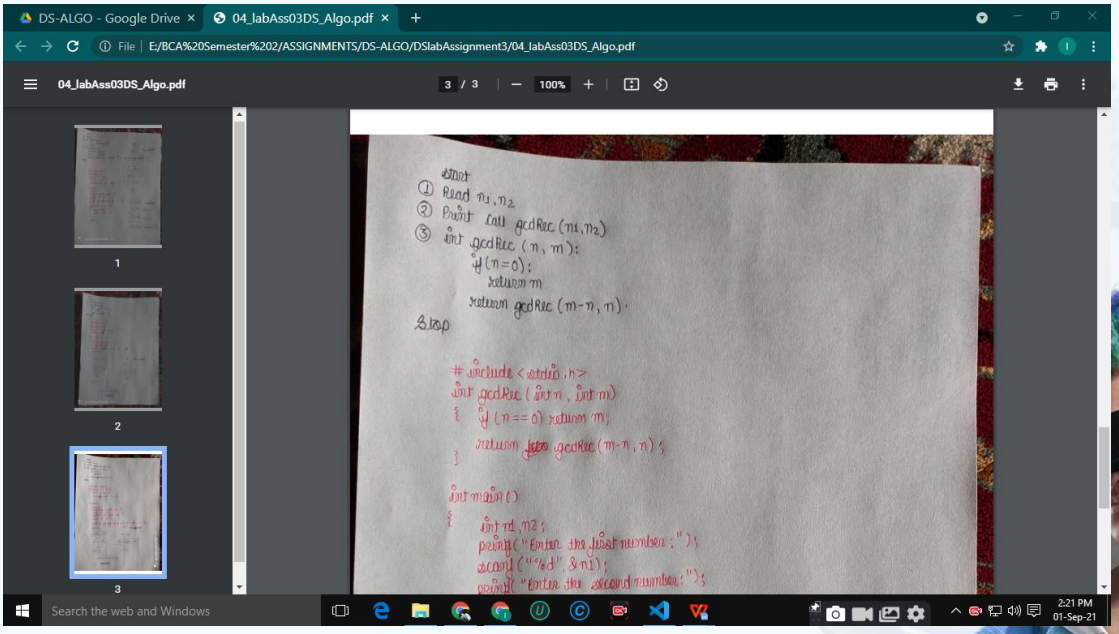
```c
#include<stdio.h>

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

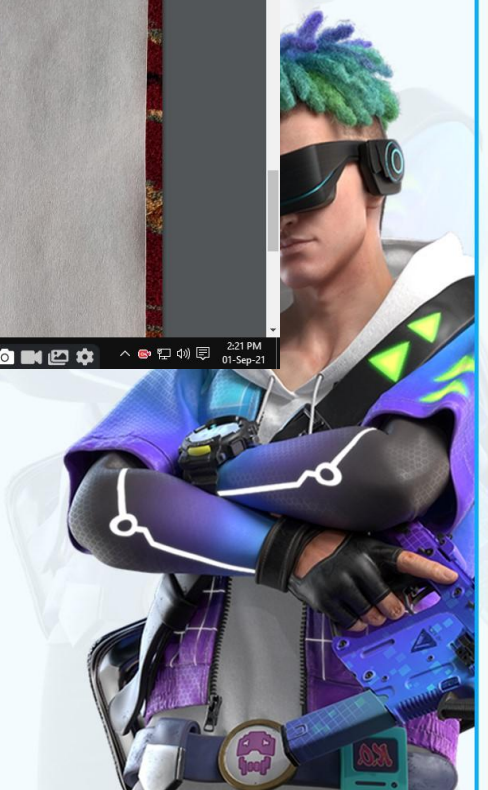
```c
int main()
{
    int n1,n2;
    printf("Enter the first number : ");
    scanf("%d", &n1);
    printf("Enter the second number : ");
    scanf("%d", &n2);
    printf("The Greatest Common Divisor of the two number is
 %d", fiboRec(n1,n2));
    // for(int i=0; i<=n; i++)
    // {
    //     int z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

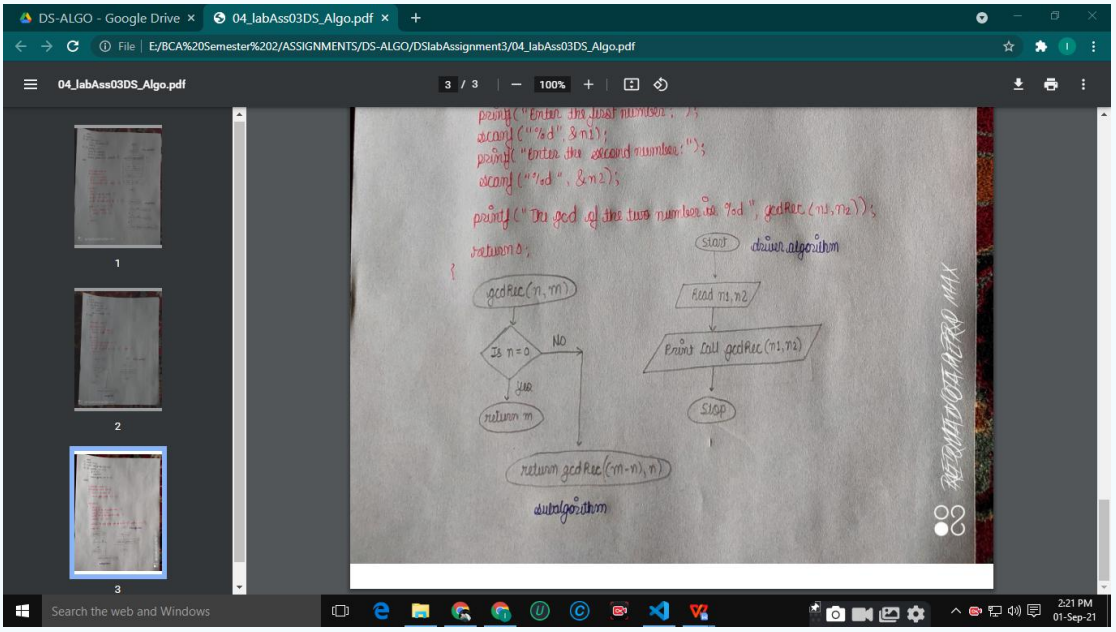


```
    // }
    return 0;
}
```


03_gcdRec.cpp - DS-ALGO - Visual Studio Code
#include <stdio.h>
int hcf(int n1, int n2);
int main()
{
int n1, n2;
printf("Enter two positive integers: ");
scanf("%d %d", &n1, &n2);
printf("G.C.D of %d and %d is %d.", n1, n2, hcf(n1, n2));
return 0;
}
int hcf(int n1, int n2)
{
if (n2 != 0)
return hcf(n2, n1 % n2);
else
Windows PowerShell
Copyright (C) 2015 Microsoft Corporation. All rights reserved.
Enter two positive integers: 4
5
G.C.D of 4 and 5 is 1.PS E:\BCA Semester 2\ASSIGNMENTS\DS-ALGO\DSlabAssignment3>


04_labAss03DS_Algo.pdf
start
Read n1, n2
Print call gcdRec (n1, n2)
int gcdRec (n, m):
if (n=0):
return m
return gcdRec (m-n, n).
Stop
#include <stdio.h>
int gcdRec (int n, int m)
if (n == 0) return m;
return gcdRec (m-n, n);
int main()
int n1, n2;
printf("Enter the first number:");
scanf("%d", &n1);
printf("Enter the second number:");




DS-ALGO - Google Drive
04_labAss03DS_Algo.pdf
E:/BCA%20Semester%202/ASSIGNMENTS/DS-ALGO/DSlabAssignment3/04_labAss03DS_Algo.pdf
04_labAss03DS_Algo.pdf
3 / 3
100%
printf("Enter the first number : ");
scanf("%d", &n1);
printf("Enter the second number: ");
scanf("%d", &n2);
printf("The gcd of the two numbers is %d", gcdRec(n1,n2));
return 0;
}
gcdRec(n,m)
Is n = 0
NO
YES
return m
return gcdRec(m-n),n)
subalgorithm
start
driver algorithm
Read n1,n2
Print call gcdRec(n1,n2)
stop
REFINED DATA PRO MAX
1
2
3
Search the web and Windows
2:21 PM
01-Sep-21

Lab Assignment 4

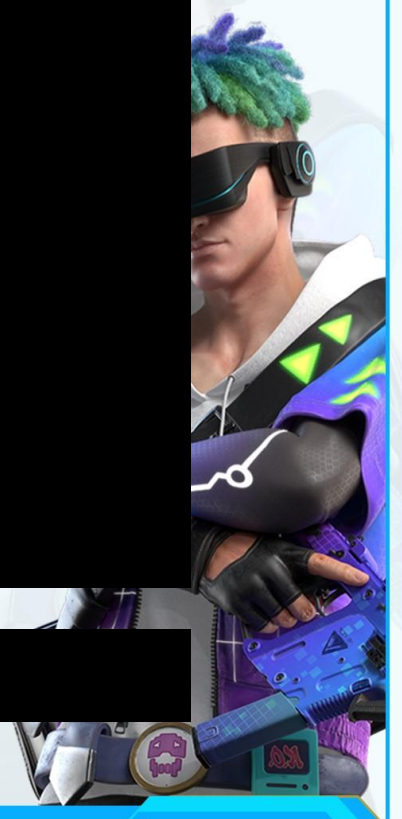
01_ArrayInsertion_StaticArray

```c
#include<stdio.h>


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

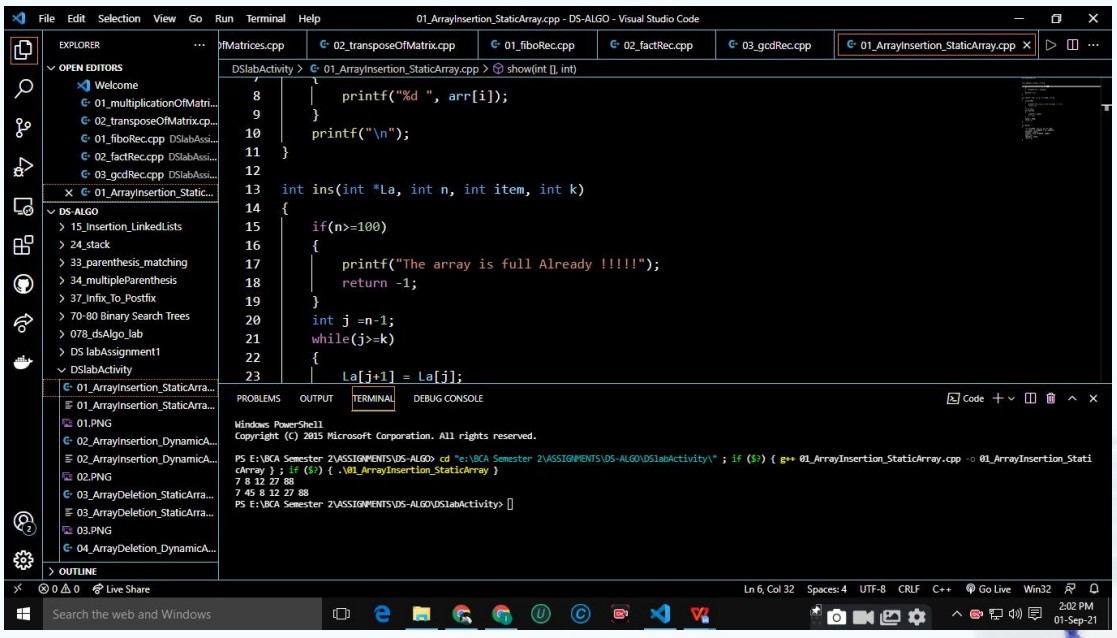
```c
    for (int i = 0; i < n; i++)
    {
        printf("%d ", arr[i]);
    }
    printf("\n");
}
```

```c
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;
}
```

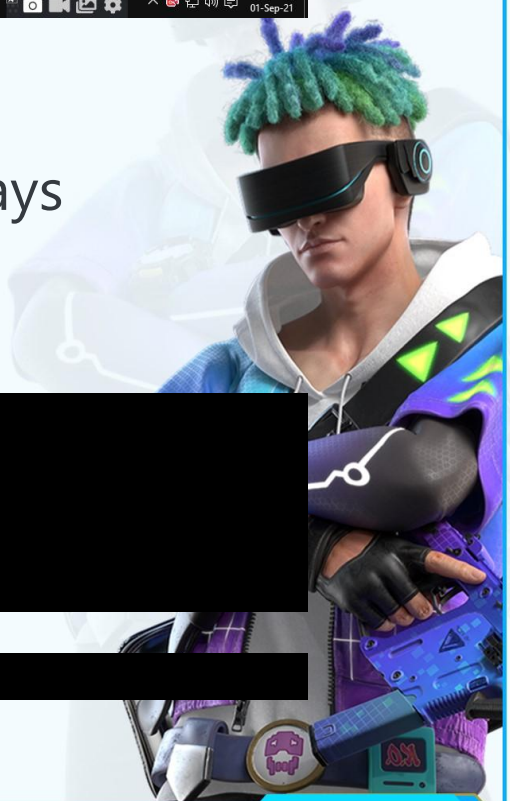
```c
int main()
{
```



```
    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;
}
```



02_ArrayInsertion_DynamicArrays



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


void show(int *arr, int n){
```

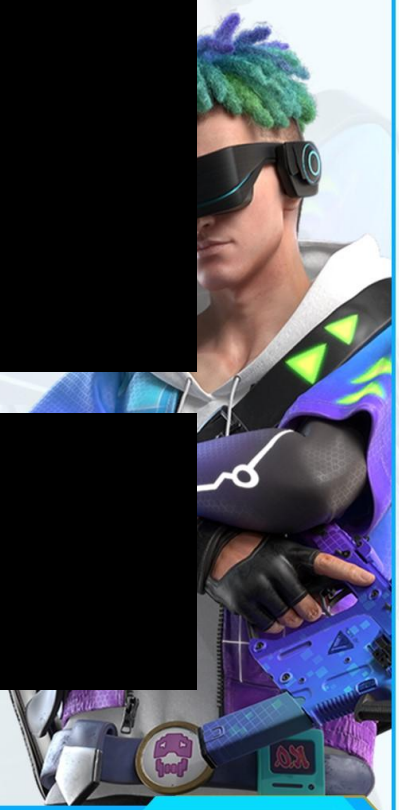
```
    for (int i = 0; i < n; i++)
```

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

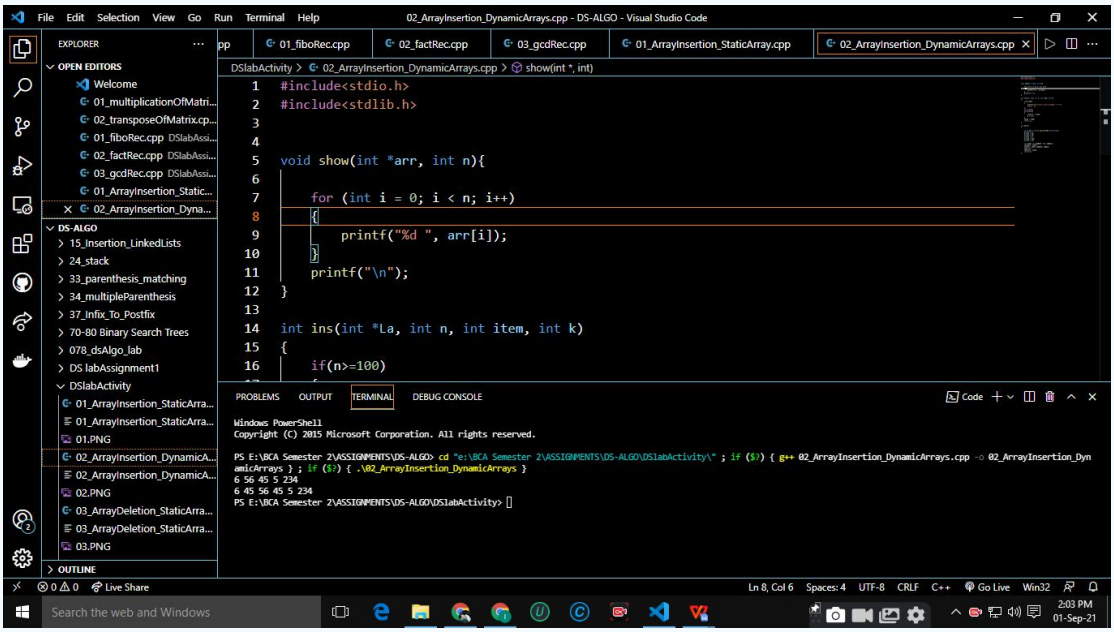
```
int ins(int *La, int n, int item, int k)
{
    if(n>=100)
    {
        printf("The array is full Already !!!!!");
        return -1;
    }
    int j =n-1;
    while(j>=k)
    {
        La[j+1] = La[j];
        j = j-1;
    }
    La[k] = item;
    return 1;
}
```

```
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

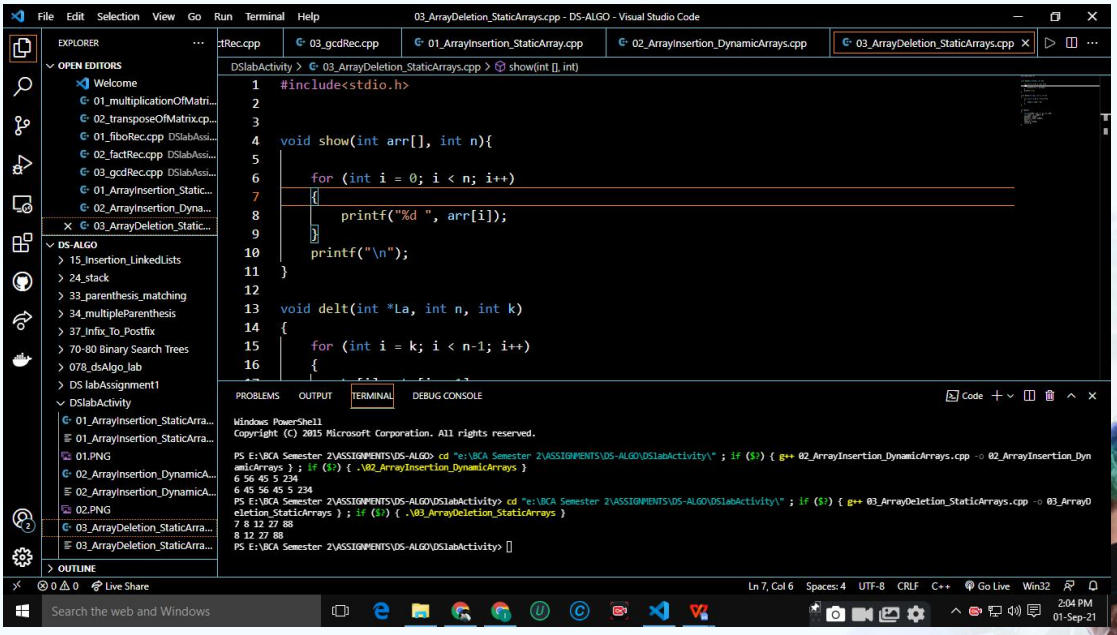
```
#include<stdio.h>


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

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

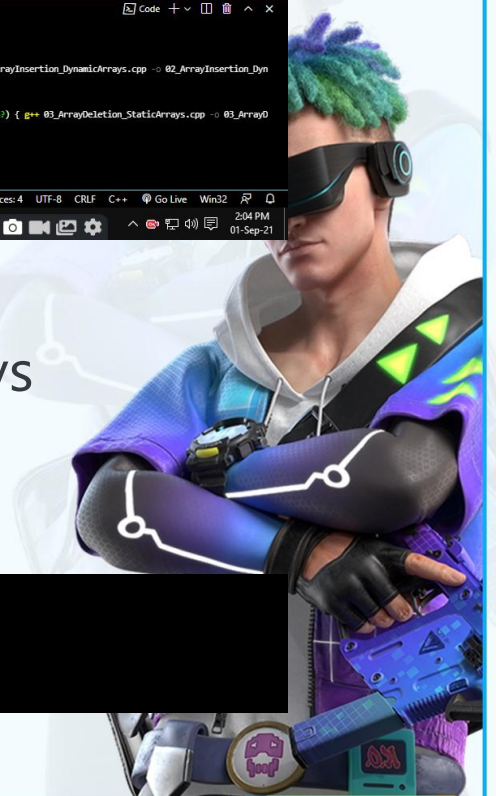
```
void delt(int *La, int n, int k)
{
    for (int i = k; i < n-1; i++)
    {
```

```
        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



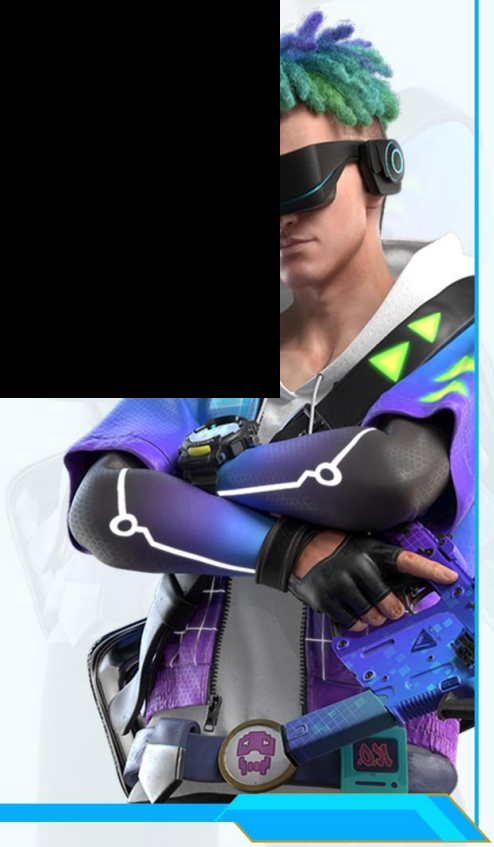
```
#include<stdio.h>
#include<stdlib.h>
```

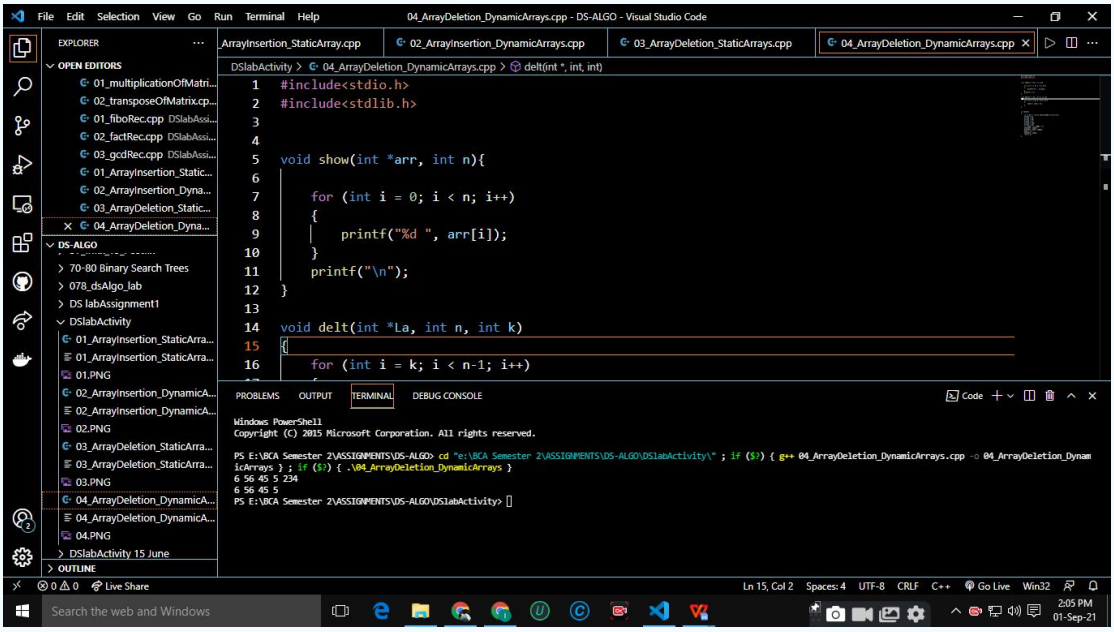
```c
void show(int *arr, int n){
```

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

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

```c
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

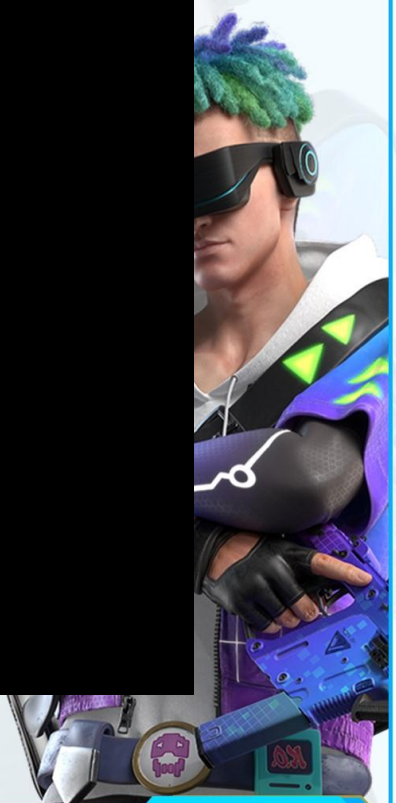
```
#include<stdio.h>
#include<string.h>
#include<stdlib.h>

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;
    }
}
```

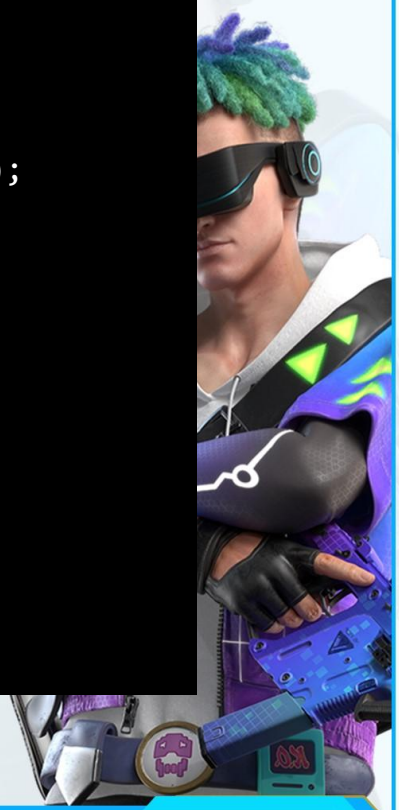
```
// deletion
void deletion()
{
    if (top == -1)
    {
        printf("\nStack is already empty !!!!\n");
        return;
    }
    else
    {
        int temp=stack[top];
        top = top - 1;
        printf("\nElement deleted is %d\n", temp);
    }
}
```

```
// 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");
    }
}
```

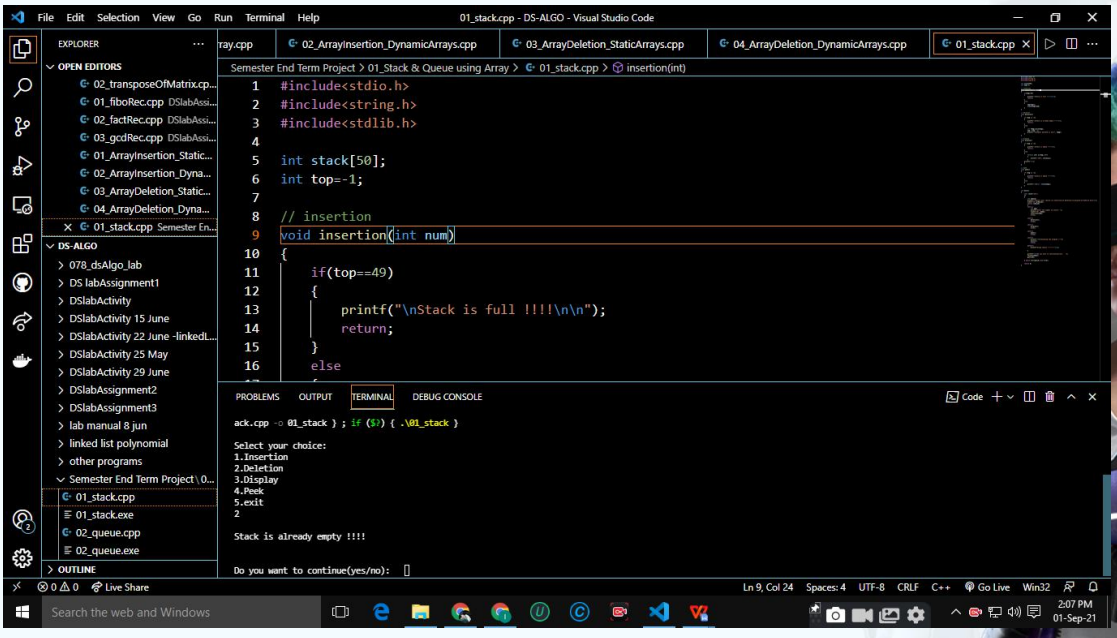


```c
// peek
void peek()
{
    if (top == -1)
    {
        printf("\nStack is empty !!!!\n");
        return;
    }
    else
    {
        printf("\n%d\n", stack[top]);
    }
}
```

```c
int main()
{
    char ch[5]="yes";
    do
    {
        int choice;
        printf("\nSelect your choice:\n1.Insertion\n2.Deleti
on\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;
}
```

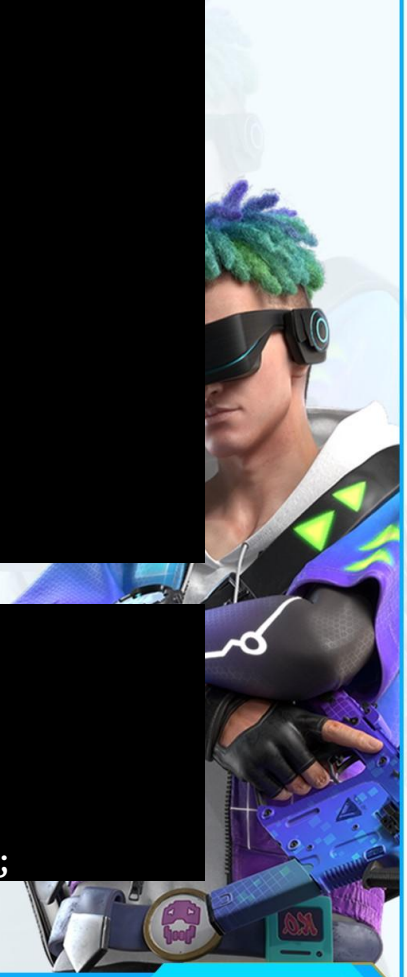


02_queue

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

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

```c
// 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;
    }
}
```

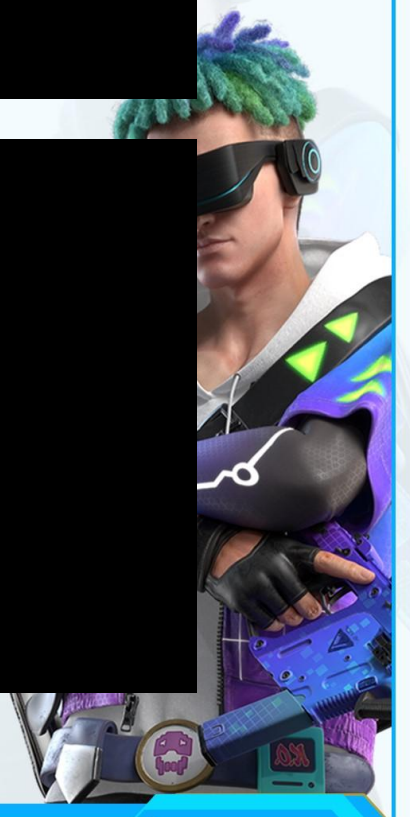
```c
// deletion
void deletion()
{
    if (front == -1)
    {
        printf("\nQueue is already empty !!!!\n");
```



```
        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.Deleti
on\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;
```

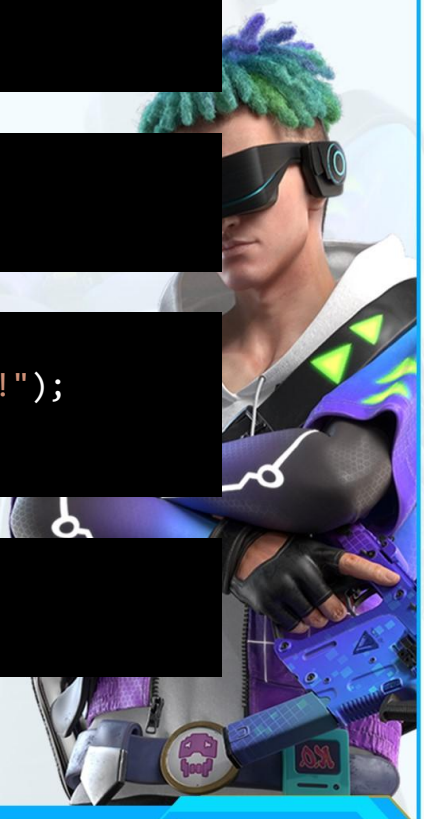
```
        case 2:
            deletion();
            break;
```

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

```
        case 4:
            peek();
            break;
```

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

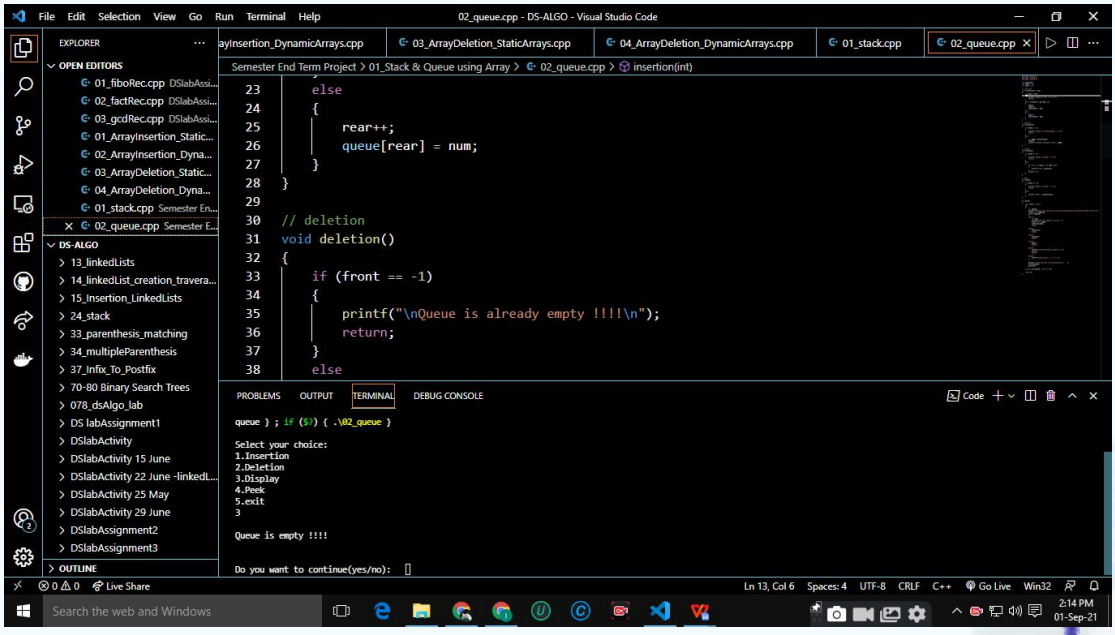
```
        default:
            printf("Wrong choice !!!!!!!!!\n");
        }
```



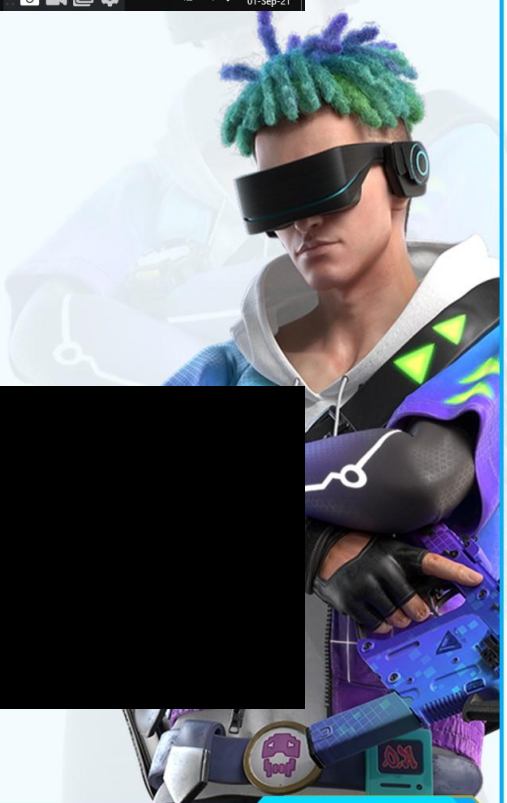
```
        printf("\n\nDo you want to continue(yes/no):    ");
        fflush(stdin);
        gets(ch);
```

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

```
    return 0;
}
```



03_circularQueue



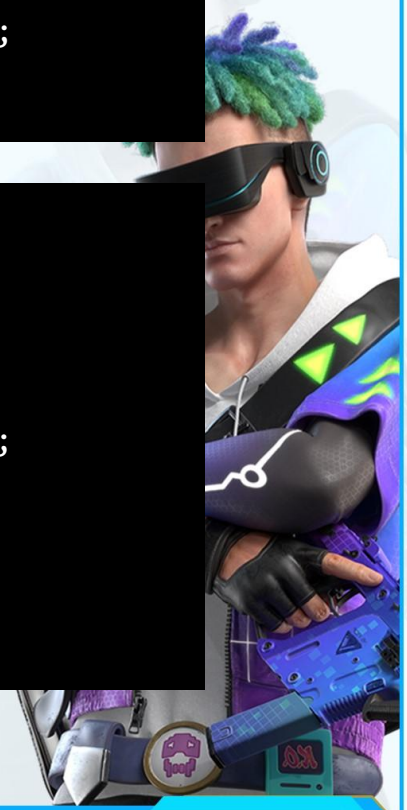
```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#define max 50

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

```c
int rear = -1;
```

```c
// 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;

}
```

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

```c
    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);
    }
}
```

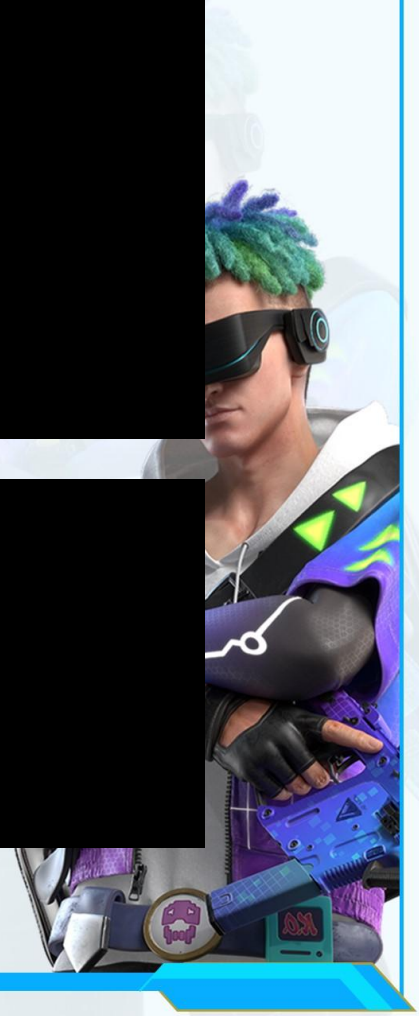
```
// display
void display()
{
    printf("\nFront = %d\nRear = %d\n", front, rear);
```

```
    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]);
    }
}
```

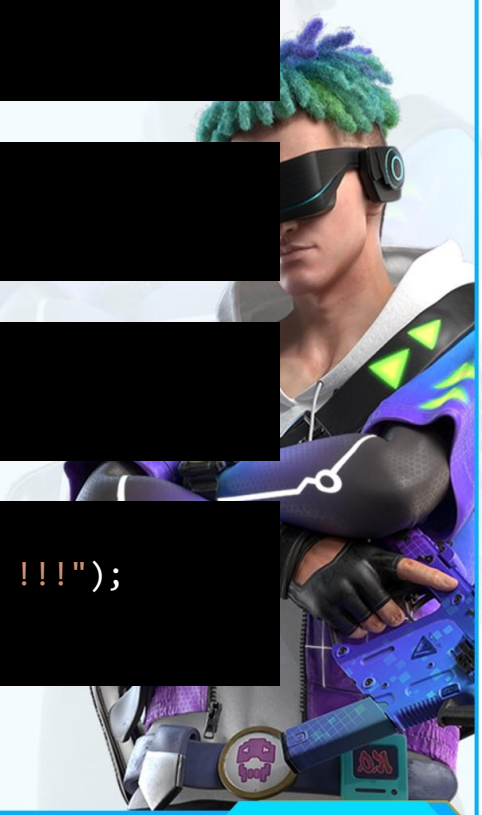
```
int main()
{
    char ch[5] = "yes";
    do
    {
        int choice;
        printf("\nSelect your choice:\n1.Insertion\n2.Deleti
on\n3.Display\n4.Peek\n5.exit\n");
        scanf("%d", &choice);
        switch (choice)
        {
        case 1:
            int num;
            printf("Enter the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

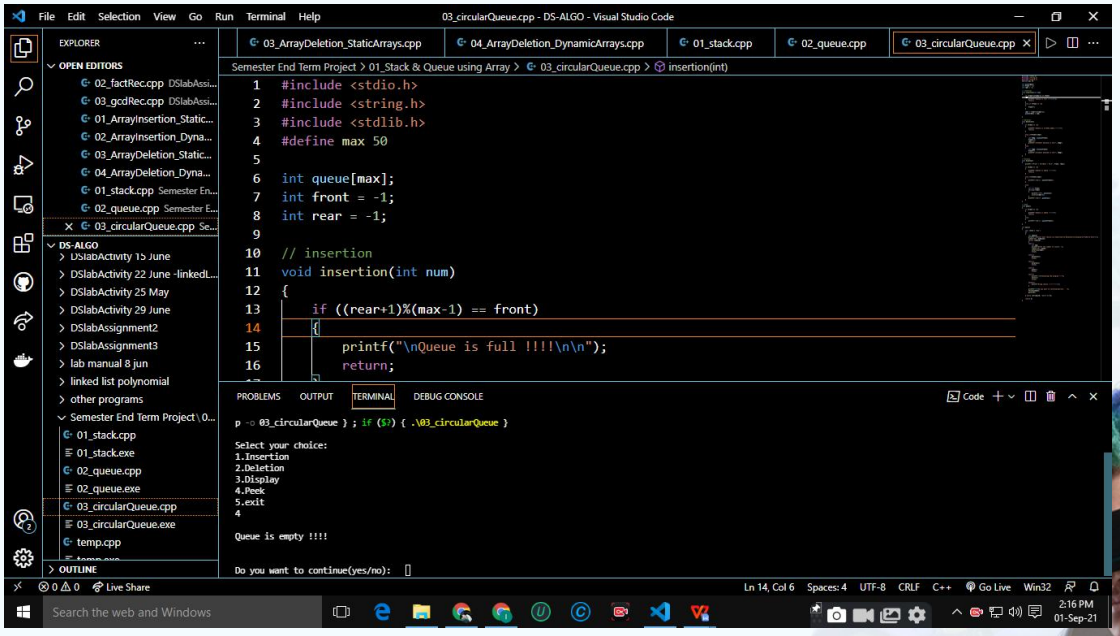


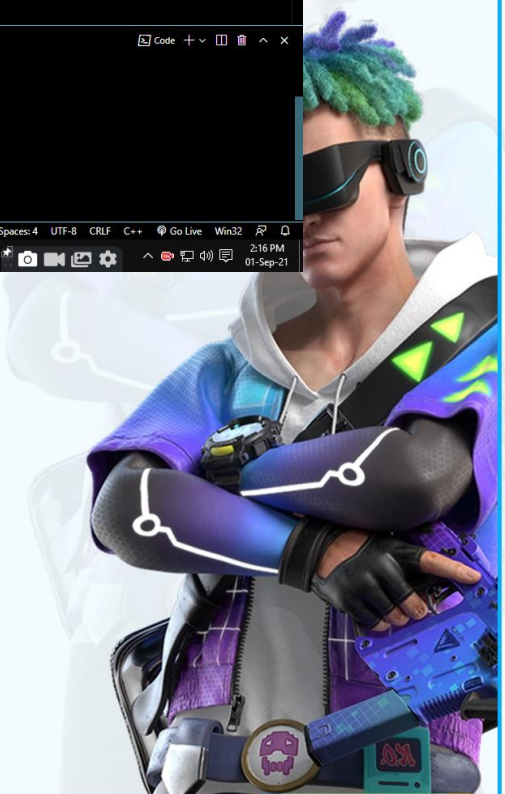
```
        default:
            printf("Wrong choice !!!!!!!!!\n");
        }
```

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

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

```
    return 0;
}
```





Lab Assignment 6

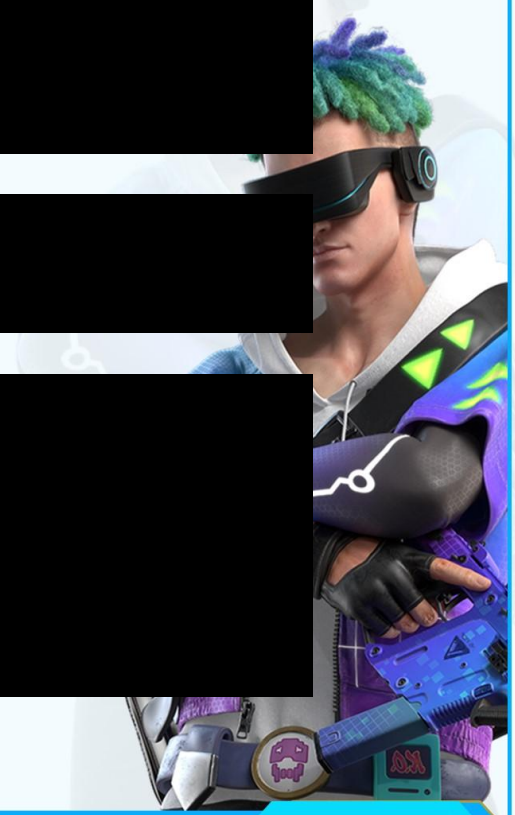
01_queueOperations

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

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

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

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

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

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



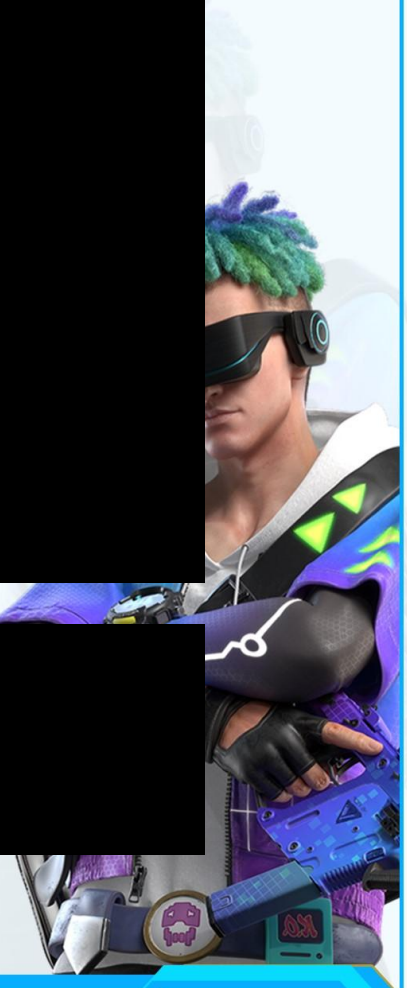
```
        if(front==rear)
        {
            front=-1;
            rear=-1;
        }
```

```
        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]);
    }
}
```

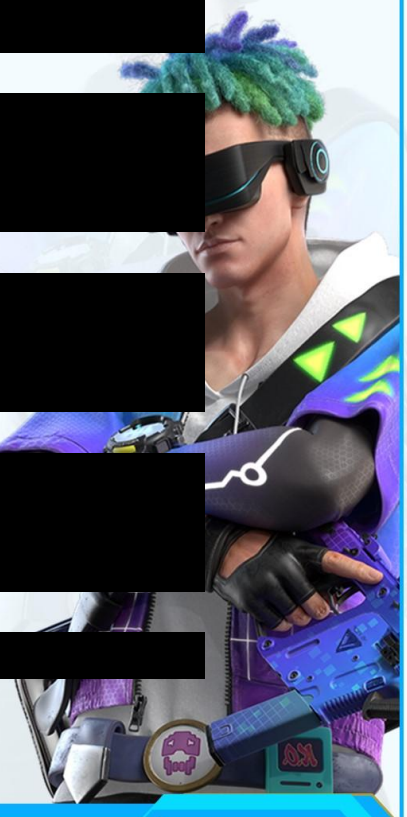
```
int main()
{
    char ch[5] = "yes";
    do
    {
        int choice;
        printf("\nSelect your choice:\n1.Insertion\n2.Deleti
on\n3.Display\n4.Peek\n5.exit\n");
        scanf("%d", &choice);
        switch (choice)
        {
        case 1:
            int num;
            printf("Enter the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

```
        case 2:
            deletion();
            break;
```

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

```
        case 4:
            peek();
            break;
```

```
        case 5:
```



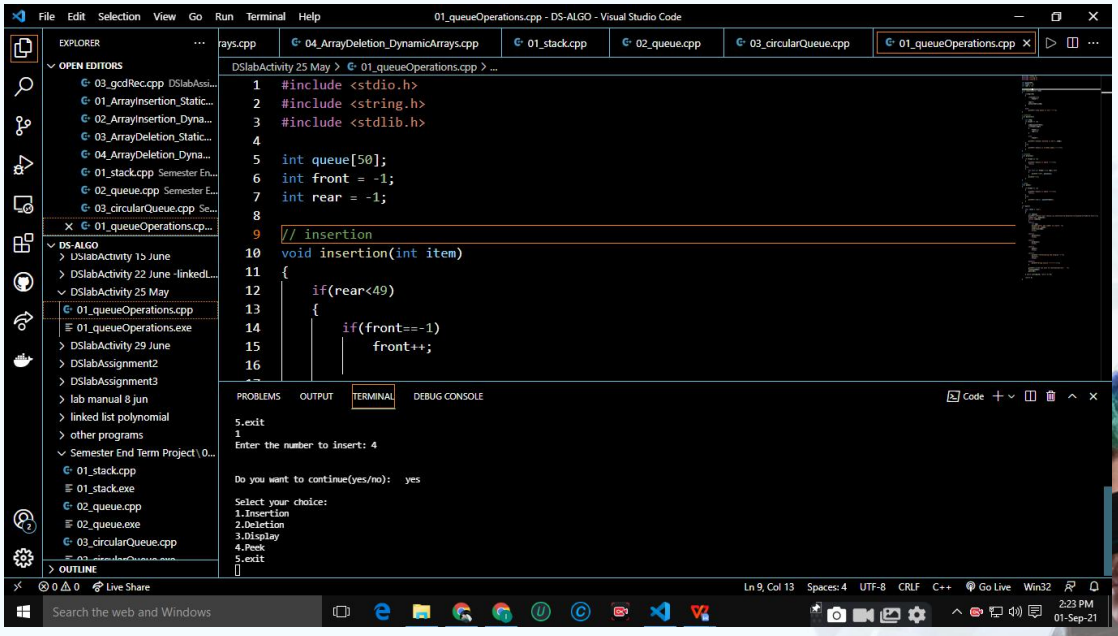
```
            printf("\nTerminating the program !!!");
            exit(1);
            break;
```

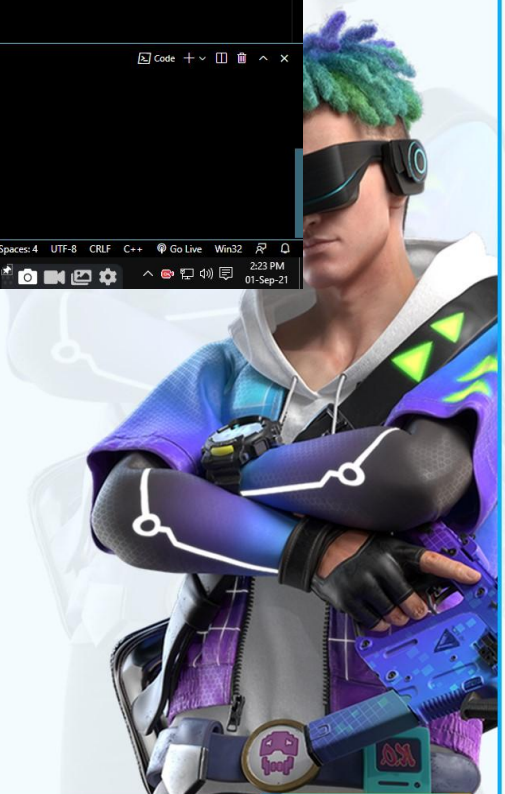
```
        default:
            printf("Wrong choice !!!!!!!!!\n");
        }
```

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

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

```
    return 0;
}
```





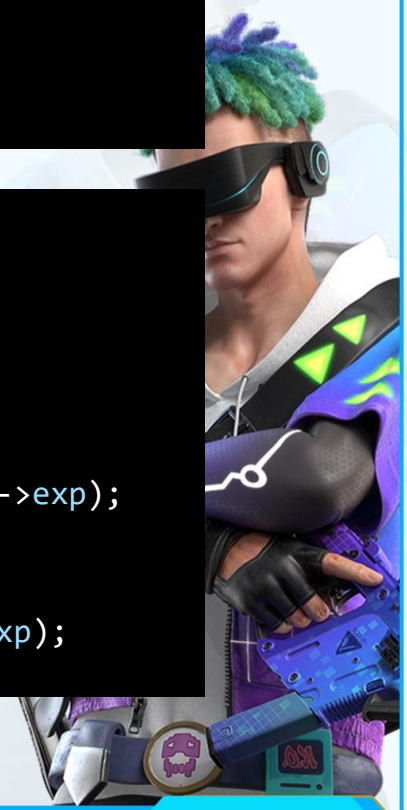
Lab Assignment 7

01_linkedList_polynomial

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

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

```c
// funtion 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;
}
```

```c
// 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);
}
```

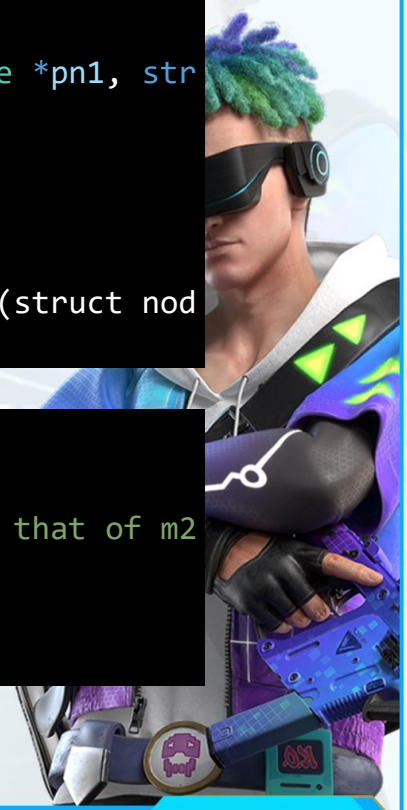


```c
// function for insertion
struct node *insertion(struct node *head, int ex, float cof)
{
    struct node *temp = (struct node *)malloc(sizeof(struct
node));
    // printf("%4.2fx^%d\n", cof, ex);
    temp->coef = cof;
    temp->exp = ex;
    temp->next = NULL;
    // printf("%4.2fx^%d\n", temp->coef,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

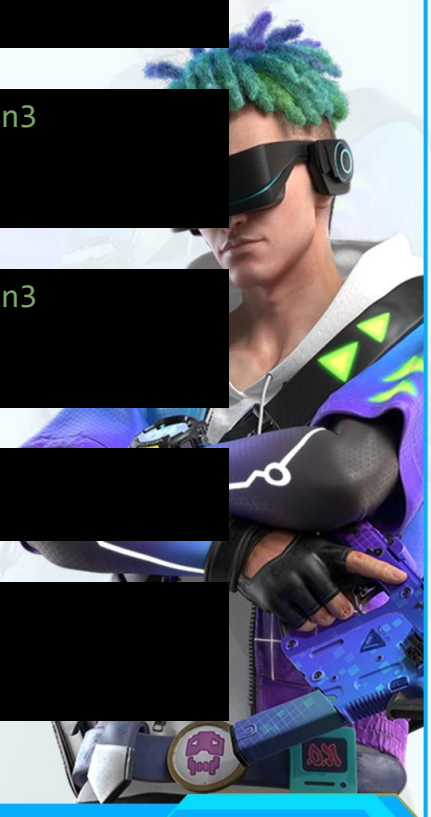
```c
        ptr->next = temp;
    }
    return head;
}
```

```c
// 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));
```

```c
    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';
```



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

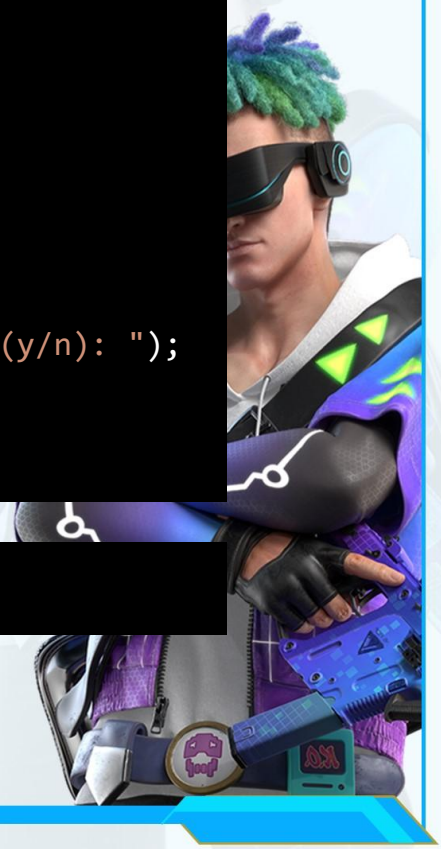
```c
    printf("Polynomial 1: \n");
    while (ch == 'y')
    {
        int ex;
        float cof;
        printf("Enter Coefficient : ");
        scanf("%f", &cof);
        printf("Enter Exponent : ");
        scanf("%d", &ex);
        pn1 = insertion(pn1, ex, cof);
        printf("Do you want to insert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

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

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

```c
    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);
    }
```

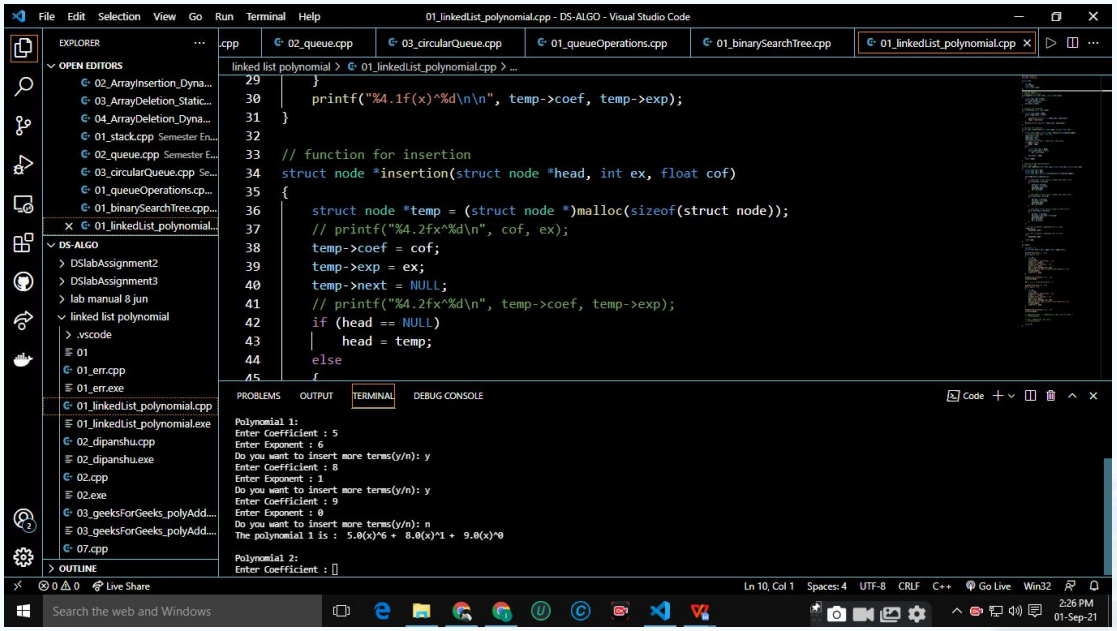
```c
    printf("The polynomial 2 is : ");
    traversal(pn2);
```



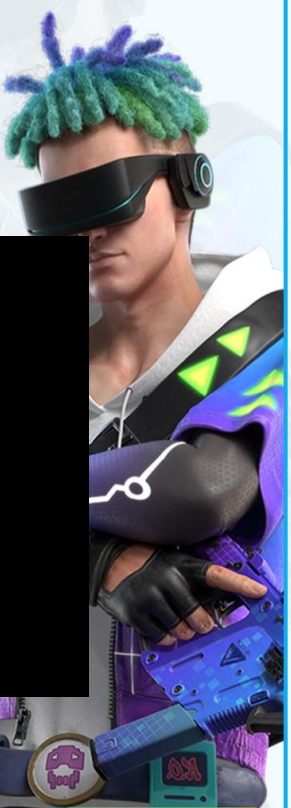
```
    // append(pn1,pn2);  // append pn1 to pn2, just for test
 :)
    // traversal(pn2);
```

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

```
    return 0;
}
```



03_geeksForGeeks_polyAdd

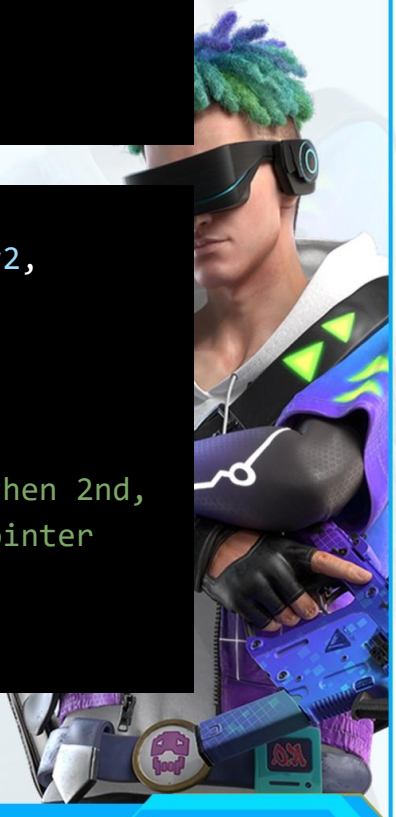


```
#include <bits/stdc++.h>
using namespace std;

// 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;
```

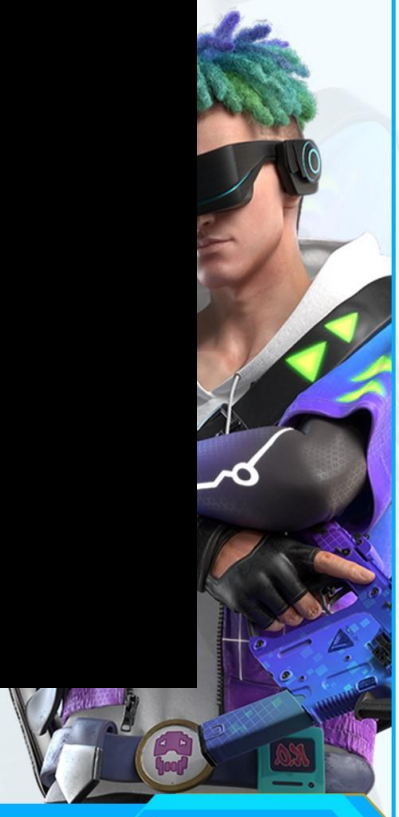


```
            poly->coeff = poly1->coeff;
            poly1 = poly1->next;
        }
```

```
        // 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 Nod
e));
        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;
```



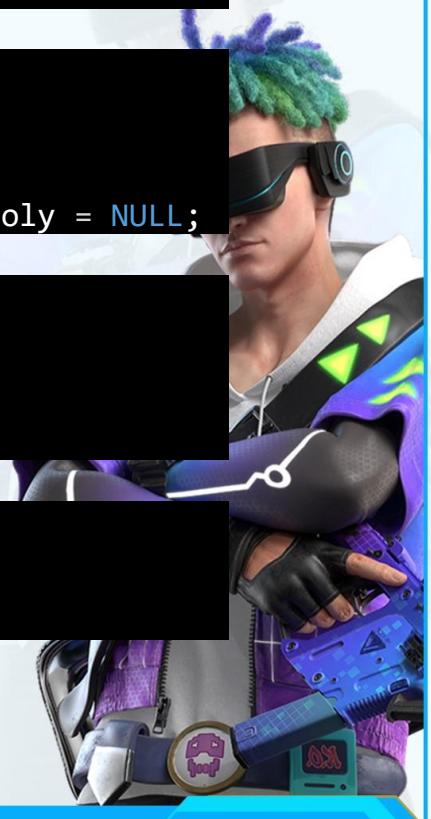
```
            poly->coeff = poly2->coeff;
            poly2 = poly2->next;
        }
        poly->next = (struct Node *)malloc(sizeof(struct Nod
e));
        poly = poly->next;
        poly->next = NULL;
    }
}
```

```
// 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");
}
```

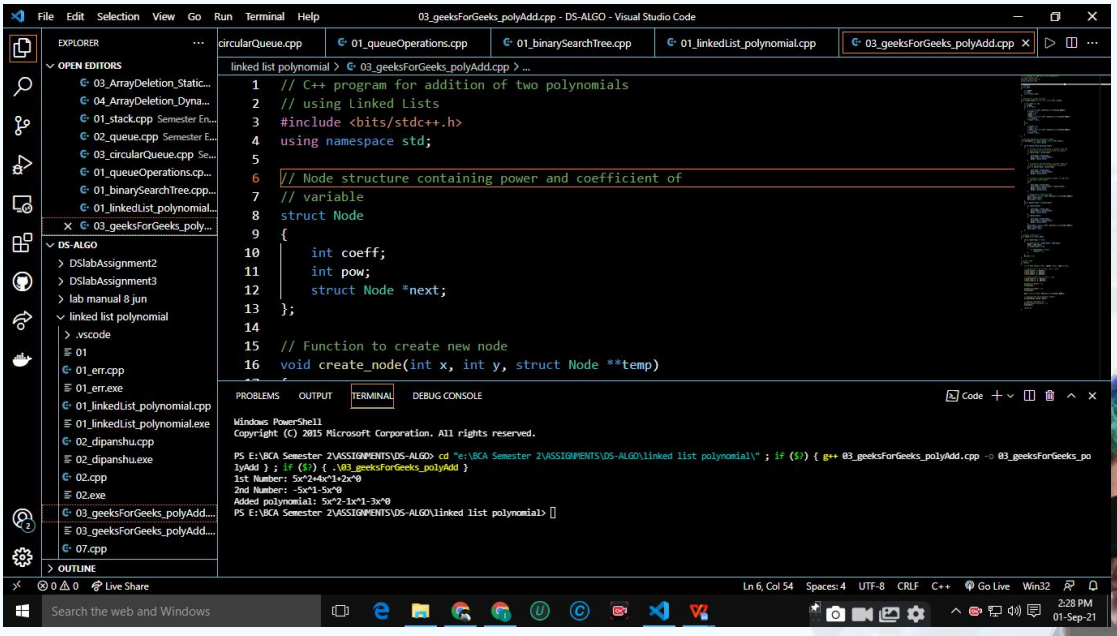
```
// Driver code
int main()
{
    struct Node *poly1 = NULL, *poly2 = NULL, *poly = NULL;
```

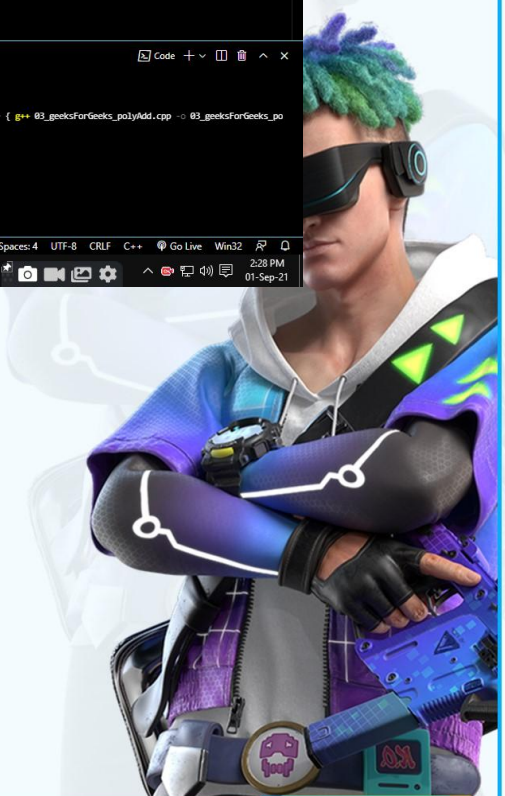
```
    // Create first list of 5x^2 + 4x^1 + 2x^0
    create_node(5, 2, &poly1);
    create_node(4, 1, &poly1);
    create_node(2, 0, &poly1);
```

```
    // 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;
}
```





Lab Assignment 8

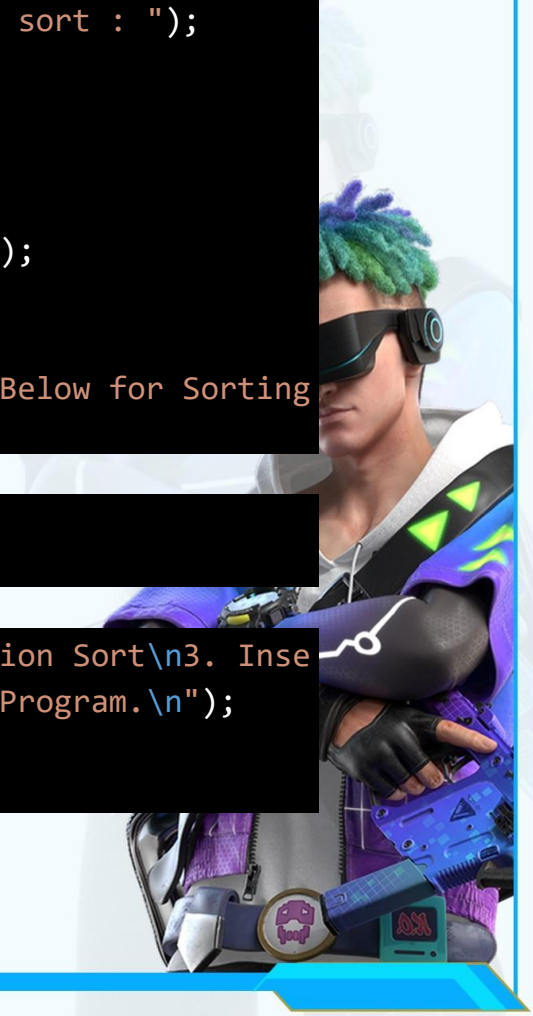
01_sorting

```c
#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);
```

```c
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");
```

```c
    while (1)
    {
```

```c
        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);
```



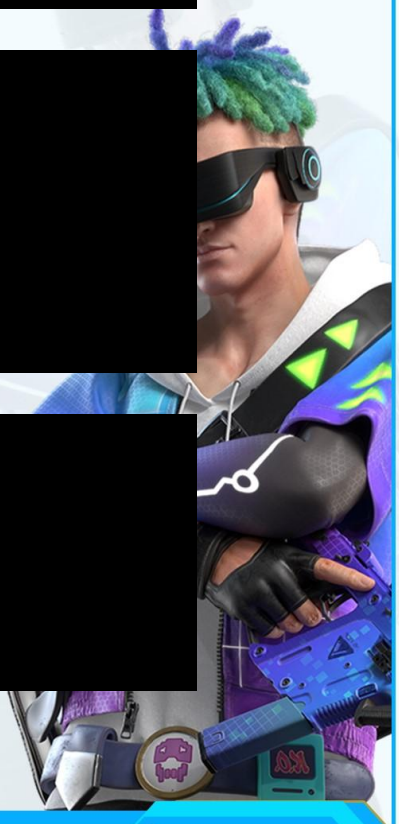
```
        switch (choice)
        {
        case 1:
            bubble_sort(arr, n);
            break;
        case 2:
            selection_sort(arr, n);
            break;
        case 3:
            insertion_sort(arr, n);
            break;
        case 4:
```

```
            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]);
    }
}
```

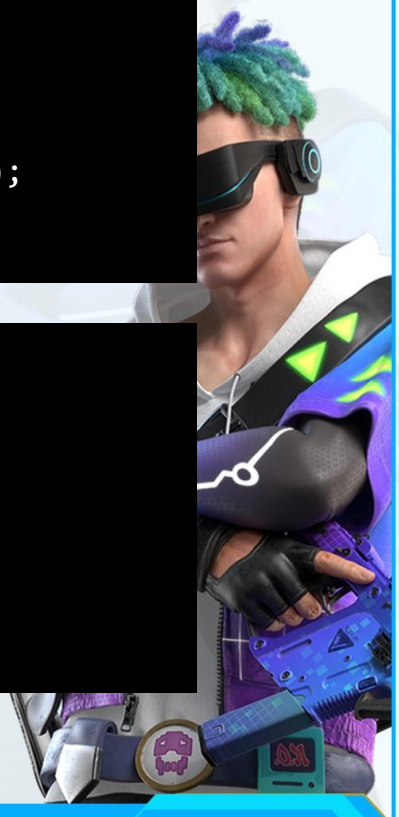
```
void bubble_sort(int arr[], int n)
{
    int i, j, temp;
    for (i = 0; i < n; i++)
    {
        for (j = 0; j < n - i - 1; j++)
```



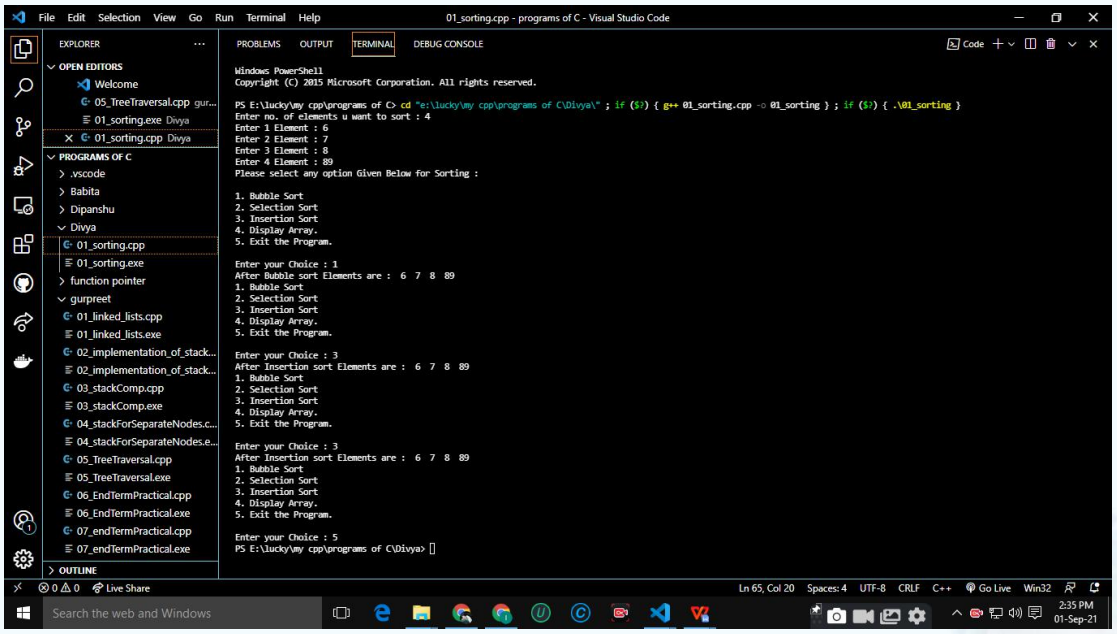
```
        {
            if (arr[j] > arr[j + 1])
            {
                temp = arr[j];
                arr[j] = arr[j + 1];
                arr[j + 1] = temp;
            }
        }
    }
    printf("After Bubble sort Elements are : ");
    display(arr, n);
}
```

```
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);
}
```



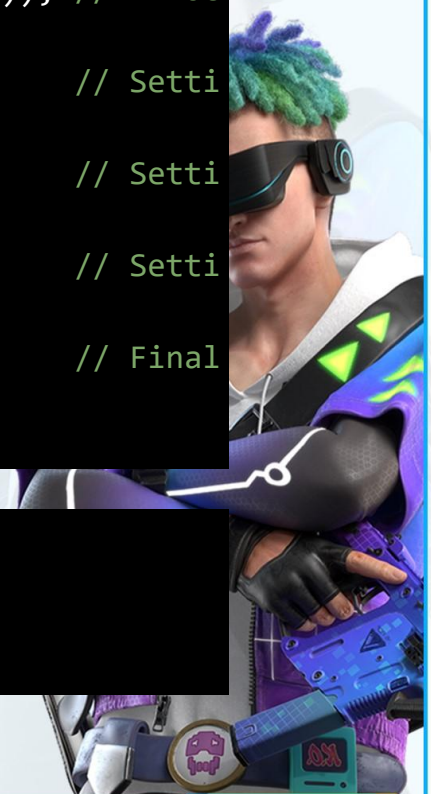
Lab Assignment 9

01_binarySearchTree

```c
// checking whether a tree is a BST?
#include <stdio.h>
#include <malloc.h>

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

```c
struct node *createNode(int data)
{
    struct node *n;                                 // creat
ing a node pointer
    n = (struct node *)malloc(sizeof(struct node)); // Alloc
ating memory in the heap
    n->data = data;                                 // Setti
ng the data
    n->left = NULL;                                 // Setti
ng the left and right children to NULL
    n->right = NULL;                                // Setti
ng the left and right children to NULL
    return n;                                       // Final
ly returning the created node
}
```

```c
void preOrder(struct node *root)
{
    if (root != NULL)
    {
```

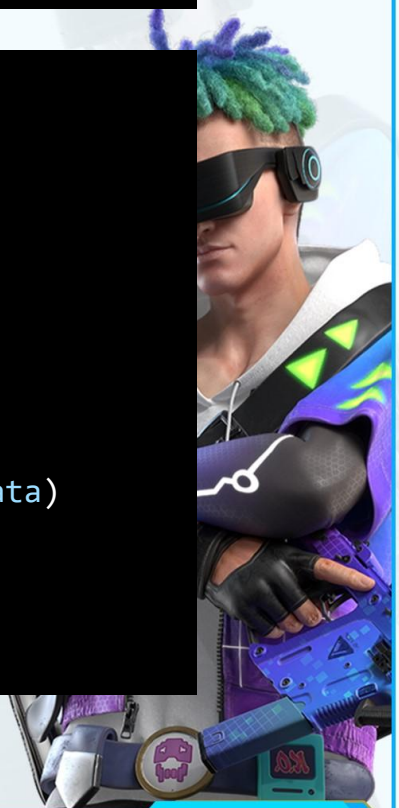


```
        printf("%d ", root->data);
        preOrder(root->left);
        preOrder(root->right);
    }
}
```

```
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;
```



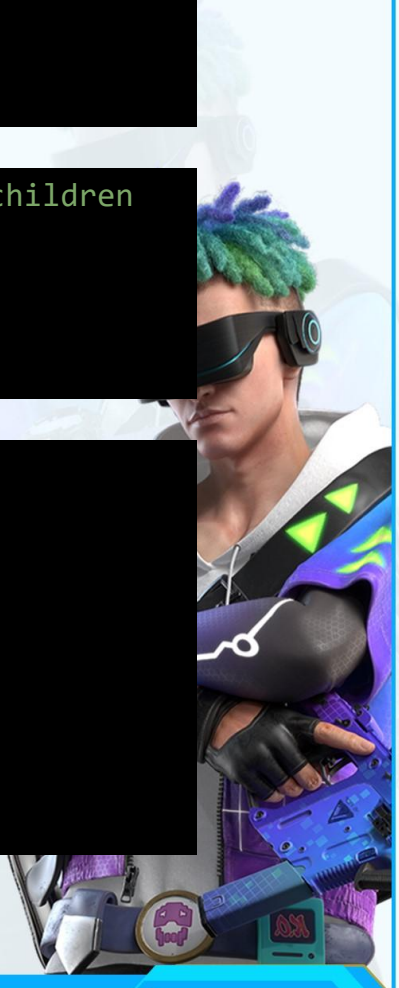
```
        return isBST(root->right);
    }
    else
    {
        return 1;
    }
}
```

```
int main()
{
```

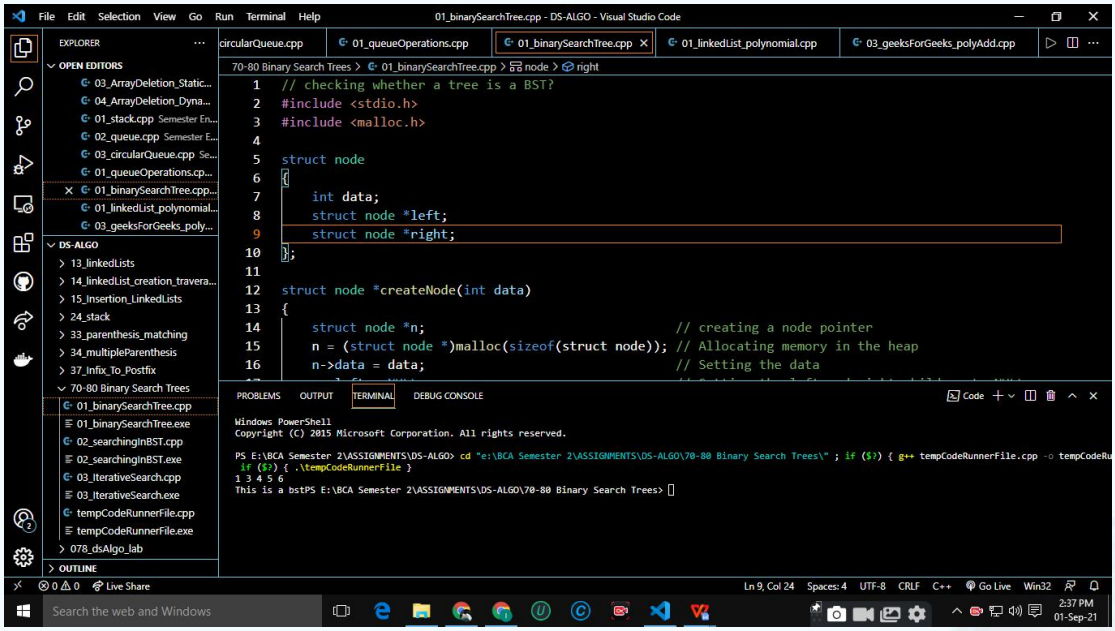
```
    // Constructing the root node - Using Function (Recommen
ded)
    struct node *p = createNode(5);
    struct node *p1 = createNode(3);
    struct node *p2 = createNode(6);
    struct node *p3 = createNode(1);
    struct node *p4 = createNode(4);
    // Finally The tree looks like this:
    //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



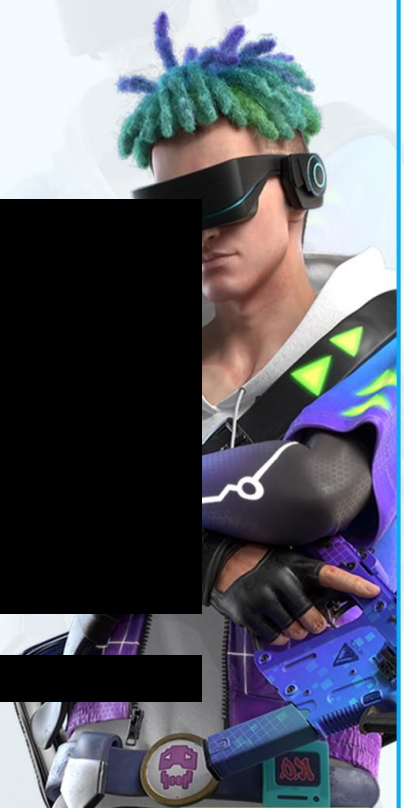
```
        printf("This is a bst");
    }
    else
    {
        printf("This is not a bst");
    }
    return 0;
}
```



02_searchingInBST

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

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

```
struct node *createNode(int data)
```

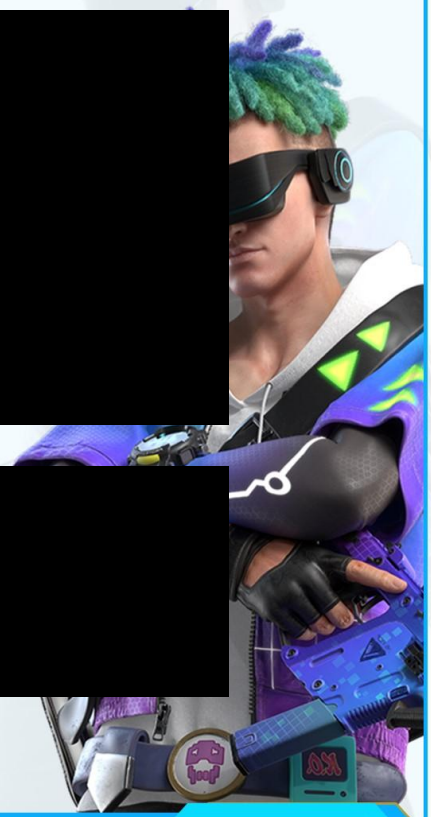


```
{
    struct node *n;                                 // creat
ing a node pointer
    n = (struct node *)malloc(sizeof(struct node)); // Alloc
ating memory in the heap
    n->data = data;                                 // Setti
ng the data
    n->left = NULL;                                 // Setti
ng the left and right children to NULL
    n->right = NULL;                                // Setti
ng the left and right children to NULL
    return n;                                       // Final
ly 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);
    }
}
```

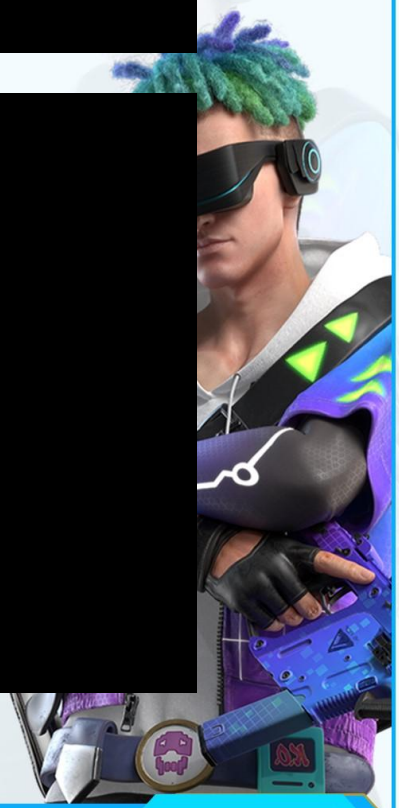
```
void inOrder(struct node *root)
{
    if (root != NULL)
    {
        inOrder(root->left);
```



```
        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);
```



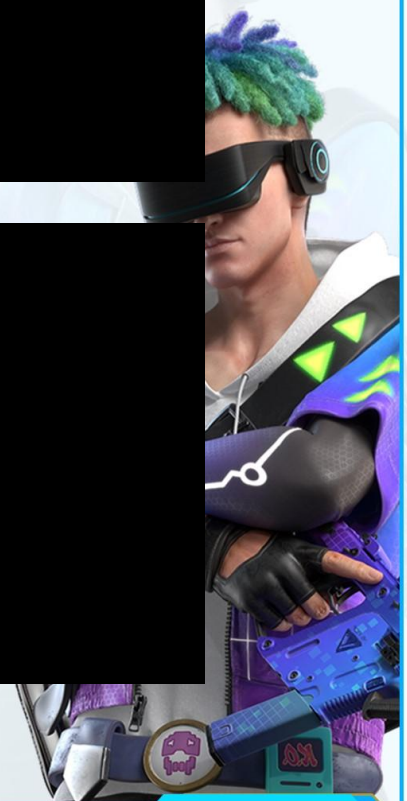
```
    }
    else
    {
        return search(root->right, key);
    }
}
```

```
int main()
{
```

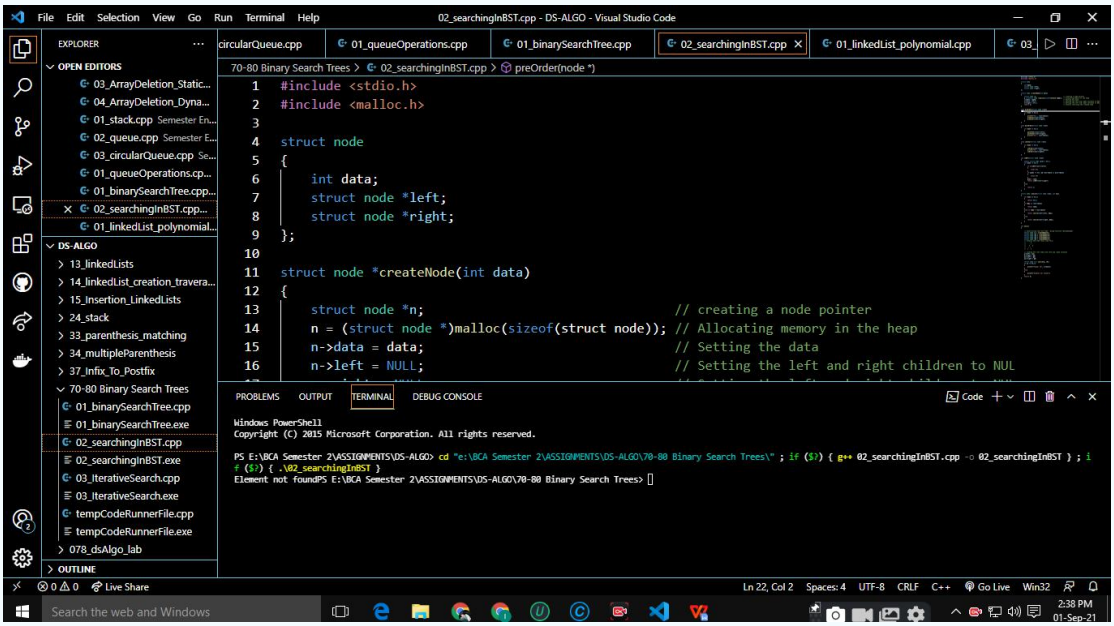
```
    // Constructing the root node - Using Function (Recommen
ded)
    struct node *p = createNode(5);
    struct node *p1 = createNode(3);
    struct node *p2 = createNode(6);
    struct node *p3 = createNode(1);
    struct node *p4 = createNode(4);
    // Finally The tree looks like this:
    //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

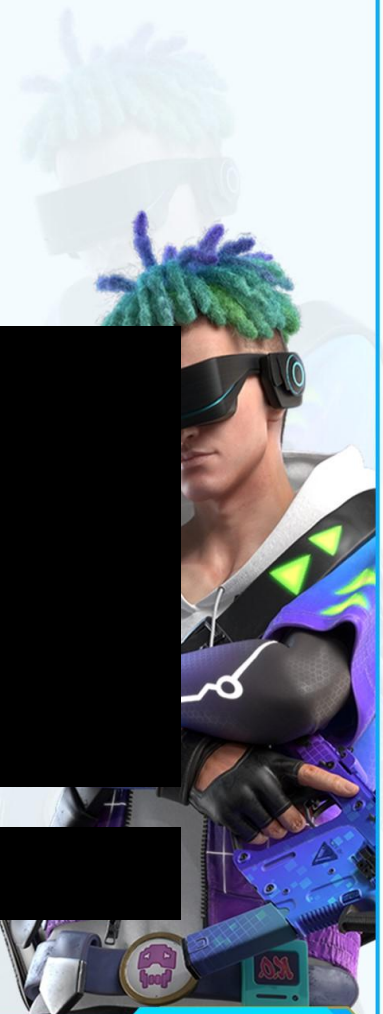
```
    struct node *n = search(p, 10);
    if (n != NULL)
    {
        printf("Found: %d", n->data);
    }
    else
    {
        printf("Element not found");
    }
    return 0;
```



```
}
```



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;
```

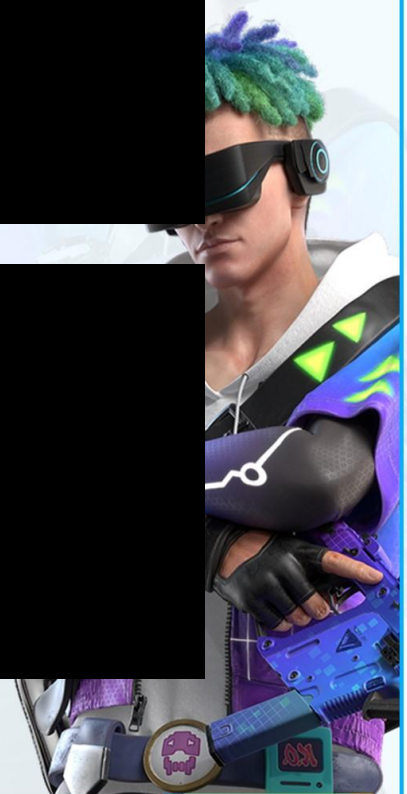
```
struct node *create(int num)
{
```

```
    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);
    }
}
```

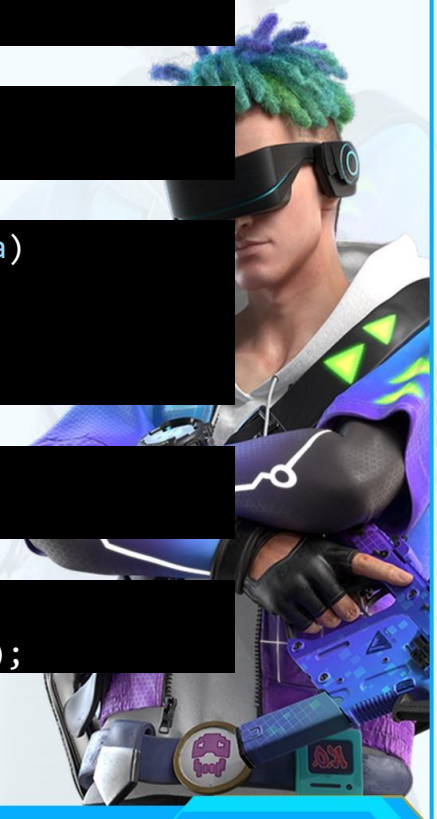


```c
void inorder(struct node *root)
{
    if (root != NULL)
    {
        inorder(root->left);
        printf("%d\n", root->data);
        inorder(root->right);
    }
}
```

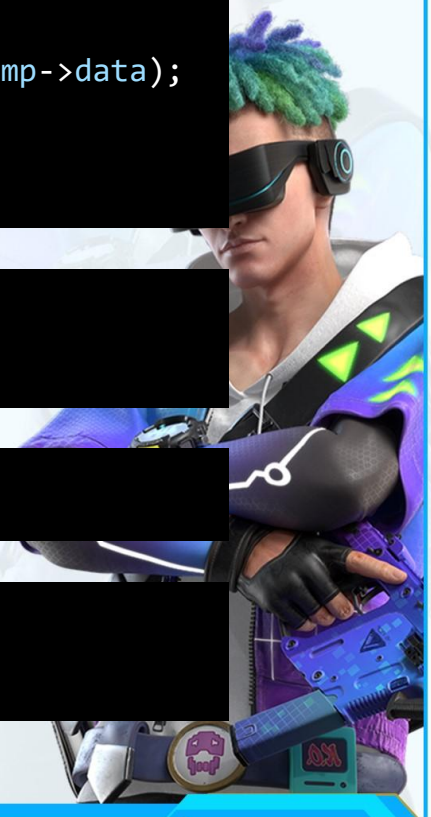
```c
void postorder(struct node *root)
{
    if (root != NULL)
    {
        postorder(root->left);
        postorder(root->right);
        printf("%d\n", root->data);
    }
}
```

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

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

    return current;
}
```

```c
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;
    }
}
```

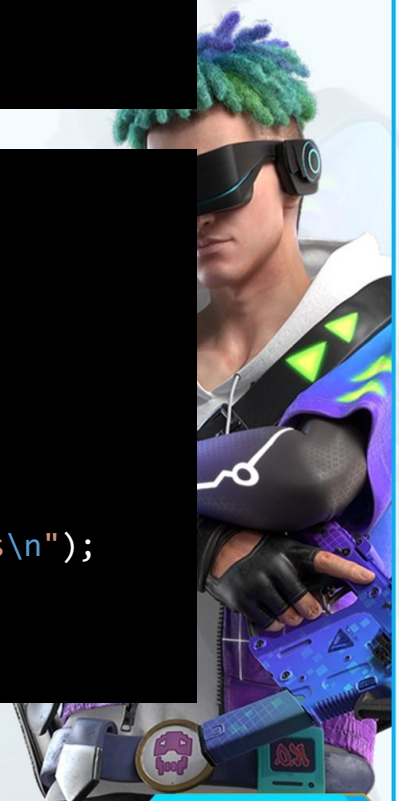
```
void deleteTree(struct node *root)
{
    if (root == NULL)
        return;
```

```
    /* 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;
```

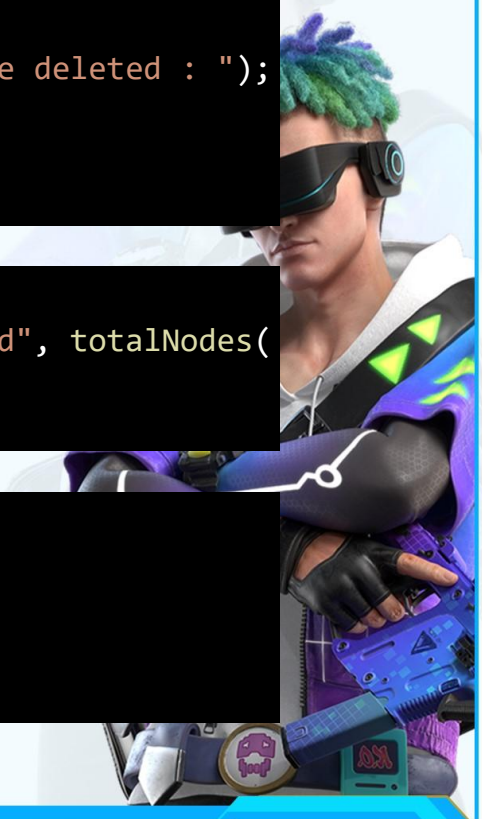
```
        case 3:
            printf("\n The elements of the tree are : \n");
            inorder(root);
            break;
```

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

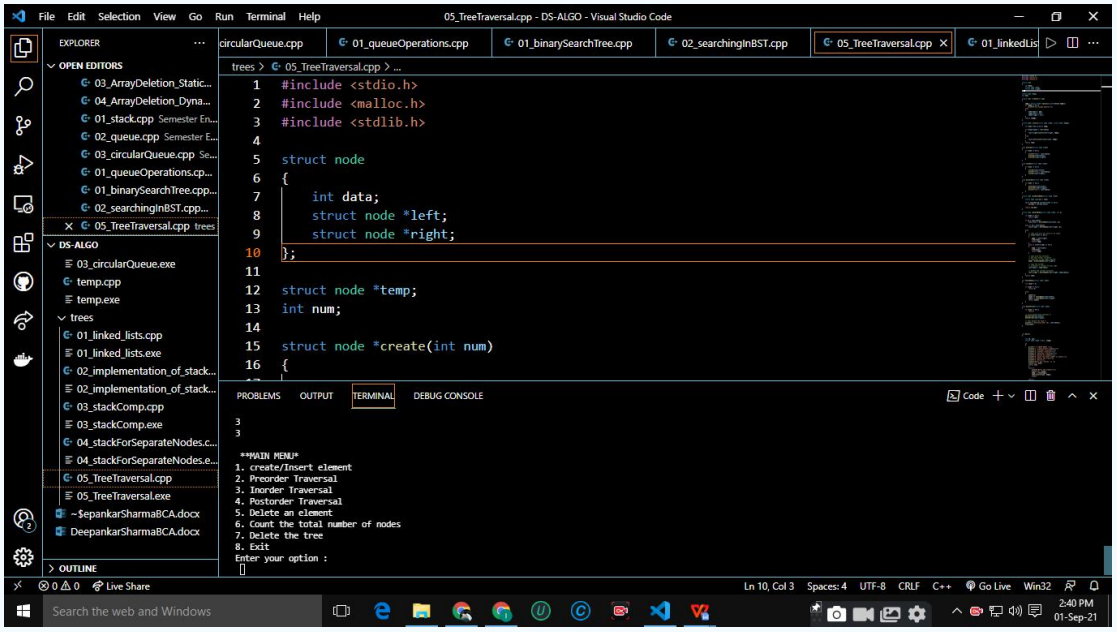
```
        case 5:
            printf("\n Enter the element to be deleted : ");
            scanf("%d", &val);
            root = deleteNode(root, val);
            break;
```

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

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



```
    return 0;
}
```



Rest of the lab practicals are at : here