NOTRE DAME UNIVERSITY BANGLADESH

Data Structure Report

Task-01

Submitted to: Humayara Binte Rashid

Submitted by: Shazidul Alam

Subject: Data Structure

Student ID: 0692220005101009

Batch: CSE-19

Task : 1

CSE-19

ID: 0692220005101009

1.Write a program in C to store N numbers of elements in an array and print it.

```
#include <stdio.h>
int main() {
   int N, i;
printf("Enter no. of elements: ");
   scanf("%d", &N);
   int array[N];
printf("Enter the elements:\n");
   for (i = 0; i < N; i++) {
      printf("A[%d] = ", i);
      scanf("%d", &array[i]); }
printf("\nArray is:\n");
   for (i = 0; i < N; i++) {
      printf("A[%d] = %d\n", i, array[i]);  }
   return 0;}
```

1) Let, array size = N

At first, we are taking an input in the variable N and then, we are turning it to the size of the array. The process is described below:

Suppose, the value of N is 3.

When $i = 0$; $0 < 3 \rightarrow$ true $\rightarrow$ Input will be taken. in array [0], let's say the input is $i++ \Rightarrow 0 + 1 = 1$; $1 < 3 \rightarrow$ true $\rightarrow$ Input is taken in array[1], let's say the input is 10.

$i++ \Rightarrow 1 + 1 = 2$; $2 < 3 \rightarrow$ true $\rightarrow$ Input is taken in array [2], If the input is 20, $i++ \Rightarrow 2 + 3 = 3$; $3 < 3 \rightarrow$ False

Loop ended.

Printing the array: A [0] = 5

A [1] = 10

A [2] = 20

CSE-19

ID:0692220005101009

2. Write a program in C to find out the even values of an array.

```
#include <stdio.h>
int main() {
   int N, i;
printf("Enter no. of elements: ");
   scanf("%d", &N);
   int array[N];
printf("Enter the elements:\n");
   for (i = 0; i < N; i++) {
      printf("A[%d] = ", i);
      scanf("%d", &array[i]);   }
printf("\nEven values in the array:\n");
   for (i = 0; i < N; i++) {
      if (array[i] % 2 == 0) {
         printf("%d\n", array[i]);
      }}
   return 0;}
```

2) Let, array size = N.
At first, we are taking an input in the variable N and then, we are passing it to the size of the array. The process:
Suppose, the value of N is 3.
When i=0; 0<3 → true → Input will be taken in array[0], let's say the input is i++ => 0+1=1; 1<3 → true → Input is taken in array[1], let's say the input is 10.
i++ => 1+1=2; 2<3 → true → Input is taken in array[2]. If the input is 20, i++ => array[2]. If the input is 20, i++ => 2+2=3<3 → False.
Loop ended.
Now, another for loop will start for finding the even values and printing them.
When i=0; 0<3 → true → Input is already taken.
Now, it will go through "if" condition,
[if(array[0]%2==0)] → true → Then, the array will print.
i++ => 0+1=1; 1<3 → true → Input is already taken.
[if (array[1]%2==0)] → true → there the array will print i++ => 1+1=2; 2<3 → true → Input is already taken now, ~~if~~
[if (array[2]%2=0)] → true → then then array will print. i++ => 2+1=3; 3<3 → False. Loop Ended.
The inputs that pass through if condition will print.

CSE-19

ID: 0192220005101009

3. Write a program in C to find out the values of the odd indexes of N numbers ofelements of an array.

```
#include <stdio.h>
int main() {
   int N, i;
printf("Enter no. of elements: ");
   scanf("%d", &N);
   int array[N];
printf("Enter the elements:\n");
   for (i = 1; i <= N; i++) {
      printf("A[%d] = ", i);
      scanf("%d", &array[i]);   }
printf("\nValues at odd indexes:\n");
   for (i = 1; i <= N; i += 2) {
printf("%d\n", array[i]);   }
   return 0;}
```

3) Array size = N.
At first, we are taking an input in the variable N and then, we are turning it into the size of the array. Process:
Suppose, the value of N is 3.
When i=0; 0<3 → true → Input will be taken in array [0], let's say, the input is 5. i++ => 0+1=1; 1<3 → true → Input is taken in array [1], let's say the input is 10. i++ => 1+1=2; 2<3 → true → Input is taken in array [2], let's say the input is 20. i++ => 2+1=3; 3<3 → False. Loop ended.
Now, we will start another "for loop" for printing the odd indexes only. The process:
When i=0; 0<3 → true → Input will be taken in array [0], let's say, the input is 5. i = i+2 => 0+2 = 2; 2<3 → true → Input will be taken in array [2], let's say the input is 20. i = i+2 => Since there are no elements at array [4]. ∴ Loop ended.
Putting the array: A[0] = 5
A[2] = 20

CSE-17

ID:0692220005101009

4. Write a program in C to find the sum of all elements of the array of N elements.

```
#include <stdio.h>
int main() {
   int N, i;
   float sum = 0;
printf("Enter no. of elements: ");
   scanf("%d", &N);
   float array[N];
printf("Enter the elements:\n");
   for (i = 0; i < N; i++) {
      printf("A[%d] = ", i);
      scanf("%f", &array[i]);
      sum += array[i];   }
printf("\nSum of %d elements is %.2f\n", N, sum);
   return 0;}
```

4) On a floating array with array size N be float array[N] and a variable sum = 0. At first, we are taking an input in the variable N and then, we are turning it to the size of the array.

The process:

The value of N is 3.

When i = 0; 0<3 → true → Input will be taken in array[0], let's say the input is 5. Now, add the value in sum variable i.e. sum = sum + array[0]. i++ => 0+1=1; 1<3 → true → Input is taken in array[1]. Let's say the input is 10, Now, add the value in sum variable, sum = sum + array[1]. i++ => 1+1=2; 2<3 → true → Input is taken in array[2]. Let's say the input is 20, Now, we add the value in sum variable, sum = sum + array[2] i++ =>

= 2+1=3; 3<3 → False.

Loop ended.

CSE-19

ID:0692220005101009

5. Write a program in C to find the average of all elements of the array.

```
#include <stdio.h>
int main() {
   int N, i;
   float sum = 0;
   float average;
printf("Enter no. of elements: ");
   scanf("%d", &N);
   float array[N];
printf("Enter the elements:\n");
   for (i = 0; i < N; i++) {
      printf("A[%d] = ", i);
      scanf("%f", &array[i]);
      sum += array[i];   }
   average = sum / N;
 printf("\nAverage of %d elements is %.2f\n", N, average);
   return 0;}
```

5) A floating array size N be float array [N], a variable sum = 0 and a float variable average.
At first, we are taking an input in the variable N and then, we are turning it to the size of the array. The process:
The value of N is 3. When i = 0; 0<3 → true → Input will be taken in array [0]. Let's say the input is 5.
Now, we add the value in sum variable.
sum = sum + array [0]. i++ => 0+1; 1<3 → true → Input is taken in array [1]. If the input is 10. Now, add the value in sum variable.
sum = sum + array [1]. i++ => 1+1 = 2; 2<3 → true → Input in taken in array [2]. If the input is 20,. Now, add the value in sum variable.
sum = sum + array [2]. i++ => 2+1 = 3; 3<3 → False.
Loop Ended.
Finally the value of sum after the loop is ended will divide by the array size and the value will store in average variable.

CSE-17

ID: 0692220005101009

6. Find out the intersection values of 2 arrays.

```
#include <stdio.h>
int main() {
   int N1, N2, i, j, k;
printf("Enter no. of elements of A: ");
   scanf("%d", &N1);
   int A[N1];
printf("Enter the elements of A:\n");
   for (i = 0; i < N1; i++) {
   scanf("%d", &A[i]); }
printf("\nEnter no. of elements of B: ");
   scanf("%d", &N2);
   int B[N2];
printf("Enter the elements of B:\n");
   for (i = 0; i < N2; i++) {
      scanf("%d", &B[i]); }
printf("\nIntersection: ");
   for (i = 0; i < N1; i++) {
   for (j = 0; j < N2; j++) {
      if (A[i] == B[j]) {
printf("%d ", A[i]);
   break;    } } }
printf("\n");
   return 0;}
```

6) Array 1 is an array and array 2 is an array with size N. I.e N[1] and B[N2].
At first, we are taking an input in the variable N and then, we are turning it to the size of ~~the~~ both of the arrays. The process:
The value of N is 3.
When i = 0; 0 < 3 → true → Input will be taken in A[0].
If the input is 5 i++ => 0 + 1 = 1,
1 < 3 → true → Input will be taken in A[1].
If the input is 10. i++ => 1 + 1 = 2;
2 < 3 → true → Input will be taken in A[2].
If the input is 20. i++ => 2 + 1 = 3;
3 < 3 → False. Loop Ended
Then, we will take input on array 2[N].
Now, another boom loop will be created for finding intersection through a nested loop. The process:
When i = 0; i < N1 → True → j = 0; j < N2 → True →
It will go through if condition.
[if (A[0] == B[0])] → True.
Then the element will print. j++ =>
i = 0; i < N1 → True → j = 0 + 1 = 1; j < N2 → True →

CSE-19

Id: 0692220005101009

Now if condition. [if (A[0] == B[1])] → True.

Then, the element will print. j++ =>

$i = 0$; $i < N_1$ → True → $j = 1 + 1 = 2$; $j < N_2$ → True →

Then, the element will print. j++ =>

$i = 0$; $i < N1$ → True → $j = 2 + 1 = 3$; $j < N2$ → False →

Then i++ => $i = 0 + 1 = 1$; $i < N_1$ → True → $j = 0$; $j < N2$ → True →

Element will print. j++ => $i = 1$; $1 < N1$ → True → $j = 0 + 1 = 1$;

$j < N2$ → True → Element will print. j++ =>

$i = 1$; $i < N1$ → True → $j = 1 + 1 = 2$; $j < N2$ → True →

Element will print j++ =>

$i = 1$; $i < N_2$ → True → $j = 2 + 1 = 3$; $j < N2$ → False →

then i++ =>

$i = 1 + 1 = 2$; $i < N_2$ → True → $j = 0$; $j < N2$ → True →

Element will print j++ =>

$i = 2$; $i < N_2$ → True → $j = 0 + 1 = 1$; $j < N2$ → True →

Element will print j++ =>

$j = 2$; $i < N2$ → True → $j = 1 + 1 = 2$; $j < N2$ → True →

Element will print j++ =>

$j = 2$; $i < N2$ → True → $j = 2 + 1 = 3$; $j < N2$ → False →

then i++ =>

$i = 2 + 1 = 3$; $i < N2$ → False. Loop End

NOTRE DAME UNIVERSITY BANGLADESH

Data Structure Report

Task-02

Submitted to: Humayara Binte Rashid

Submitted by: Shazidul Alam

Subject: Data Structure

Student ID: 0692220005101009

Batch: CSE-19

Task : 2

CSE-19
ID: 0692220005101009

1. **Insert an item into an index.**

Sample input: Enter number of elements: 5 Enter the index: 3 Enter the value to insert: 15 Enter Array: 3 28 5 11 17	Output: Array is: A[1] =3 A[2] =28 **A[3] =15** A[4] =5 A[5] =11 A[6] =17

```
#include<stdio.h>
void main(){
int i, num,pos,size;
printf("Enter number of elements :");
   scanf("%d", &size);
   int array[size+1];
   for(i=1; i<=size; i++)
{printf("Enter the %d element:\n",i);
   scanf("%d",&array[i]);}
printf("Enter the number you want to insert in the array");
   scanf("%d", &num);
printf("Enter the position of the element");
   scanf("%d", &pos);
   for(i=size; i>=pos; i--){
   array[i+1]=array[i];}
   array[pos]=num;
   size++;
   for(i=1; i<=size; i++){
printf("A[%d]=%d\n",i,array[i]);}}
```

Output:

```
Enter number of elements :5
Enter the 1 element:    3
Enter the 2 element:    28
Enter the 3 element:    5
Enter the 4 element:    11
Enter the 5 element:    17
Enter the number you want to insert in the array 15
Enter the position of the element 3
```

Simulation:

CSE-19

ID: 0692220005101009

(1)

1	2	3	4	5
3	28	5	11	17

Insert 15 at index 3.

S-1

N = 5, k = 3, item = 15

∴ k ≤ N

S-2

(a) J = N = 5

=> A[5+1] = A[5]

=> A[6] = 17

J = J - 1 = 5 - 1 = 4

1	2	3	4	5	6
3	28	5	11	17	17

(b) J = 4 ≥ k (3)

∴ A[J+1] = A[J]

=> A[4+1] = A[4]

=> A[5] = 11

J = 4 - 1 = 3

1	2	3	4	5	6
3	28	5	11	11	17

(c) J = 3 ≥ k (3)

∴ A[J+1] = A[J]

=> A[3+1] = A[3]

=> A[4] = 5

=> J = 3 - 1 = 2

1	2	3	4	5	6
3	28	15	5	11	17

(End of loop) N = N + 1

A[1]=3
A[2]=28
A[3]=15
A[4]=5
A[5]=11
A[6]=17
Process returned 6 (0x6) execution time : 44.508 s
Press any key to continue.

2. Delete an item from an index.

Sample input:	Output: Array is:
Enter number of elements: 5 Enter the index: 4 Enter Array: 3 28 5 11 17	A[1] =3 A[2] =28 A[3] =5 A[4] =17

```
#include<stdio.h>
void main(){
   int index, i, size;
printf("Enter number of elements: ");
   scanf("%d",&size);
   int a[size+1];
   for ( i = 1; i <=size; i++)
{printf("Enter the %d element :",i);
   scanf("%d",&a[i]);   }
printf("Enter the index you want to delete :");
   scanf("%d",&index);
   if (index<size)
{  for ( i = index; i <= size; i++)
   {a[i]=a[i+1]; }
   size--;
   for ( i = 1; i <= size; i++)
{printf("A[%d]=%d\n",i,a[i]); }}}
```

Output:

Enter number of elements: 5
Enter the 1 element :3

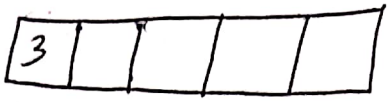
CSE-19
ID: 0692220005101009

(2)

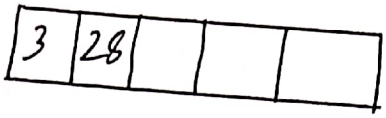
3	28	5	11	17

Se-1

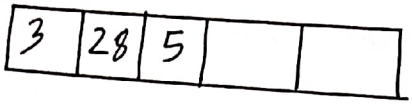
a) Set $i = 1$; $i \leq 5$
a[1] = 3
i++



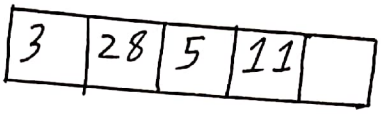
b) $i = 2$; $i \leq 5$
a[2] = 28. i++



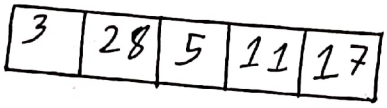
c) $i = 3$; $i \leq 5$
a[3] = 5. i++



d) $i = 4$; $i \leq 5$
a[4] = 11. i++



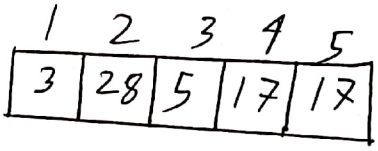
e) $i = 5$; $i \leq 5$
a[5] = 17, i++



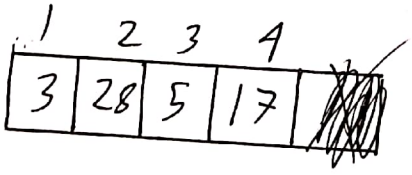
f) $i = 6$; $i \leq 5$ [False]

Se-2

a) Set $j = pos + 1$; $j \leq 5$
a[i] = a[pos+1]
=> a[i] = 17. j++



b) $j = 5$; $j < 5$ [False]
N = N - 1



```
Enter the 2 element :28
Enter the 3 element :5
Enter the 4 element :11
Enter the 5 element :17
Enter the index you want to delete :4
A[1]=3
A[2]=28
A[3]=5
A[4]=17
Process returned 4 (0x4)   execution time : 31.982 s
Press any key to continue.
```

3. *Suppose you are attending P.E class in school. You made a line randomly of* **n** numbers of students. The teacher then instructed you to stand in a line according to your height in ascending order. Here input will be the height of the students and output will be the sorted height of the students.

   Sample Input:

   Number of students **5**

   Heights: 4.8, 5.1, 6, 5.5, 4.9

   Output:

   4.8, 4.9, 5.1, 5.5, 6

```
#include <stdio.h>
void main()
{   int i, j, num;
    float array[100];
printf("Enter the number of elements for the array:\n");
    scanf("%d", &num);
    for (i = 0; i < num; i++)
    {   printf("Enter the element for index %d:\n", i + 1);
        scanf("%f", &array[i]);    }
    for (i = 0; i < num - 1; i++)
   {for (j = 0; j < num - i - 1; j++)
  {if (array[j] > array[j + 1])
  {float temp = array[j];
      array[j] = array[j + 1];
      array[j + 1] = temp; } } }
    printf("Output: Sorted Array is:\n");
    for (i = 0; i < num; i++)
{printf("A[%d] = %.1f\n", i + 1, array[i]); }}
```

CSE-19
ID: 0192220005101009

(3) 4.8, 5.1, 6, 5.5, 4.9

K-1

(a) (4.8), (5.1), 6, 5.5, 4.9

(b) ~~4.8~~, (5.1), (6), 5.5, 4.9

(c) 4.8, 5.1, (6), (5.5), 4.9
4.8, 5.1, 5.5, 6, 4.9

(d) 4.8, 5.1, 5.5, (6), (4.9)
4.8, 5.1, 5.5, 4.9, [6]

K=2

(a) (4.8), (5.1), 5.5, 4.9, [6]

(b) 4.8, (5.1), (5.5), 4.9, [6]

(c) 4.8, 5.1, (5.5), (4.9), [6]
4.8, 5.1, 4.9, [5.5, 6]

K=3

(a) (4.8), (5.1), 4.9, [5.5, 6]

(b) 4.8, (5.1), (4.9), [5.5, 6]
4.8, ~~5.1~~ 4.9, [5.1, 5.5, 6]

K=4

(a) (4.8), (4.9), [5.1, 5.5, 6]
4.8, [4.9, 5.1, 5.5, 6]

Output:

Enter the number of elements for the array: 5
Enter the element for index 1: 4.8
Enter the element for index 2: 5.1
Enter the element for index 3: 6
Enter the element for index 4: 5.5
Enter the element for index 5: 4.9
Output: Sorted Array is: A[1] = 4.8
A[2] = 4.9
A[3] = 5.1
A[4] = 5.5
A[5] = 6.0
Process returned 5 (0x5) execution time : 15.725 s
Press any key to continue.

4. Find the mth even value of an array and delete it.

Sample input:	Output:
Enter Value of m: 2 Enter the values of array: 3 28 36 11 17	The index of the 2nd even value is 36 Array is: A[1] = 3 A[2] =28 A[3] =11 A[4] =17

```
#include <stdio.h>
void main() {
  int i, n, element, j, b = 1;
  int array[100];
printf("Enter the number of elements for array:\n");
  scanf("%d", &n);
  for (i = 1; i <= n; i++) {
printf("Enter the element for %dth index:\n", i);
    scanf("%d", &array[i]); }
  for (i = 1; i <= n; i++) {
    if (array[i] % 2 == 0) {
    if (b == 2) {
  for (j = i; j < n; j++) {
   array[j] = array[j + 1]; }
 n--; } else { b++; }}}
```

```
for (i = 1; i <= n; i++) {
printf("A[%d]=%d\n", i, array[i]); }}
```

Output:

```
Enter the number of elements in the array: 5
Enter the value of m: 2
Enter the values of the array:
3
28
36
11
17
The index of the 2th even value is 36
Array is:
A[1] = 3
A[2] = 28
A[3] = 11
A[4] = 17

Process returned 0 (0x0)   execution time : 22.993 s
Press any key to continue.
```

(4)

3	28	36	11	17

S-1

a) Set, $i = 1$; $i \leq n = 5$

=> if(a[1] % 2 == 0) [False]

i++

b) $i = 2$; $i \leq n = 5$

if (a[2] % == 0) [True]

if (b == 2) [False]

b++.

i++

c) $i = 3$; $i \leq n = 5$

if (a[3] % 2 == 0) [True]

if (b == 2) True

break (loop ended)

∴ a[3] = 36.

S-2

a) Set, $j = i = 3$; $j \leq n = 5$

a[3] = a[3+1]

a[3] = a[4] = 11

j++

1	2	3	4	5
3	28	11	11	17

b) $j = 4$; $j \leq n = 5$

a[4] = a[4+1]

a[4] = a[5] = 17

j++

1	2	3	4	5
3	28	11	17	17

c) $j = 5$; $j < n = 5$ [False]

(loop ended)

n--

1	2	3	4
3	28	11	17

NOTRE DAME UNIVERSITY BANGLADESH

Data Structure

Report on Task- 03

Submitted to: HumayaraBinte Rashid

Submitted by: ShazidulAlam

Subject: Data Structure

Student ID: 069222005101009

Batch: CSE-19

Report-3

1

1.You are searching for a book in the library. Here our sorted list is the well-arranged n number of books according to unique numbers in ascending order in a shelf. Our target element is the book we prefer to read (Find the index of our desired book id assuming the 1st book is in 1 index)

Sample Input:

No. of book ids: **5**

Book IDs: 101, 102, 307, 401, 405

Find the index of Book id: **102**

Output:

The 102 book is in **2 index.**

```
#include <stdio.h>
intbinarySearch(intarr[], int n, int target)
{  int left = 1; // Start from index 1
int right = n;
while (left <= right)
{      int mid = left + (right - left) / 2;
if (arr[mid] == target)
return mid;
else if (arr[mid] < target)
left = mid + 1;
else
right = mid - 1;  }
return -1; }
int main()
{  int n, target;
printf("No. of book IDs: ");
scanf("%d", &n);
intbookIDs[n + 1]; // Increase the array size by 1
printf("Book IDs: ");
for (int i = 1; i <= n; i++)
scanf("%d", &bookIDs[i]);
printf("Find the index of Book ID: ");
scanf("%d", &target);
int index = binarySearch(bookIDs, n, target);
if (index != -1)
printf("The %d book is in %d index.\n", target, index);
else
printf("The %d book is not found.\n", target);
return 0;}
```

Output: No. of book IDs: 5
Book IDs: 101
102
307
401
405
Find the index of Book ID: 102
The 102 book is in 2 index.
Process returned 0 (0x0) execution time : 21.481 s
Press any key to continue.

(1) Searching the index of the book id.

Step-1

Set low = 1
high = 5

1	2	3	4	5
101	102	307	401	405

Step-2

Finding mid setting low = 1, max = 5

$Mid = \frac{low + high}{2} = \frac{1+6}{2} = 3$

* if (arr[mid] == target)

This condition doesn't apply.

1	2
101	102

arr[mid] = 307 > 102

high = mid - 1

Step-3

low = 1, high = 2

$Mid = \frac{low + high}{2} = \frac{1+2}{2} = 1$

2
102

if (arr[Mid] == target)

This condition doesn't apply.

arr[Mid] = 101 < 102

low = Mid + 1

Step-4

low = 2, high = 2

$Mid = \frac{low + high}{2} = \frac{2+2}{2} = 2$

if (arr[2] == target)

(item is found)

2. Apply binary search on any type of unsorted 1D array.

Sample Input: number of elements 5

3 4 5 2 11

Find: 5

Output: 2 3 4 5 11

Location: 4

```
#include <stdio.h>
intbinarySearch(int array[], int low, int high, int target) {
while (low <= high) {
int mid = low + (high - low) / 2;
if (array[mid] == target)
return mid;
else if (array[mid] < target)
low = mid + 1;
else
high = mid - 1;}
return -1;}// If the target is not found, return -1
int main() { int n;
printf("No. of elements: ");
scanf("%d", &n);
int array[n];
printf("Enter the elements:\n");
for (int i = 0; i < n; i++) {
scanf("%d", &array[i]);  }// Sorting the array in ascending order (if it's not already sorted)
for (int i = 0; i < n - 1; i++) { // Fixed the loop termination condition
for (int j = 0; j < n - i - 1; j++) {
if (array[j] > array[j + 1]) {
int temp = array[j];
array[j] = array[j + 1];
array[j + 1] = temp; }}}
int target;
printf("Find the element: ");
scanf("%d", &target);
int result = binarySearch(array, 0, n - 1, target);
if (result != -1)
printf("The %d book is in index %d\n", target, result);
else
printf("Element not found in the array.\n");
return 0;}
```

Output: No. of elements: 5
Enter the elements:
3
4
5
2
11
Find the element: 5
The 5 book is in index 3
Process returned 0 (0x0) execution time : 23.918 s
Press any key to continue.

(2) Apply binary search on any type of unsorted 1D array.

3	4	5	2	11

K=1

ⓐ (3), (4), 5, 2, 11

ⓑ 3, 4, (5), (2), 11

ⓒ ~~3, 4,~~ 3, 4, 2, 5, 11

ⓒ 3, 4, 2, (5), (11)

ⓓ 3, 4, 2, 5, [11]

K = 2

ⓐ (3), (4), 2, 5, [11]

ⓑ 3, (4), (2), 5, [11]

3, 2, 4, 5, [11]

ⓒ 3, 2, (4), (5), [11]

3, 2, 4, [5, 11]

K = 3

ⓐ (3), (2), 4, [5, 11]

2, 3, [4, 5, 11]

ⓑ 2, (3), (4), [5, 11]

2, 3, [4, 5, 11]

K = 4

ⓐ (2), (3), [4, 5, 11]

2, [3, 4, 5, 11]

Searching index of 5

Step-1:

Set low = 0, high = 4

$Mid = \frac{0+4}{2} = 2$

if (array[2] == target)

which condition doesn't apply.

array[2] = 4 < 5

low = mid + 1 = 2 + 1 = 3

Step-2:

low = 3, high = 4

$Mid = \frac{3+4}{2} = 3$

if (array[2] == target)

This condition applies.

(item is found)

3. Find out the index of the 3rd largest value in an array and insert a value in that index.

Sample Input: number of elements 5

3 4 5 2 11

Insert: 8

Output: 3 4 8 5 2 11

```
#include<stdio.h>
void main()
{  intn,i,num;
printf("Enter number of elements in the array :");
scanf("%d",&n);
intarr[n+1];
for (int i = 1; i <= n; i++)
  {scanf("%d", &arr[i]);}
intfirstLargest = 0;
intsecondLargest = 0;
intthirdLargest = 0;
intfirstLargestindex = 0;
intsecondLargestindex = 0;
intthirdLargestindex = 0;
for (int i = 1; i <= n; i++)
 {if (arr[i] >firstLargest)
     { thirdLargest = secondLargest;
secondLargest = firstLargest;
firstLargest = arr[i];
thirdLargestindex = secondLargestindex;
secondLargestindex = firstLargestindex;
firstLargestindex = i;  }
else if (arr[i] >secondLargest)
      {  thirdLargest = secondLargest;
secondLargest = arr[i];
thirdLargestindex = secondLargestindex;
secondLargestindex = i;   }
else if (arr[i] >thirdLargest)
      {  thirdLargest = arr[i];
thirdLargestindex = i;  } }
printf("Enter the data you want to insert: ");
scanf("%d", &num);
for ( i = n; i >=thirdLargestindex; i--)
{ arr[i+1]=arr[i]; }  n++;
arr[thirdLargestindex] = num;
printf("Output\n");
for ( i = 1; i <=n; i++)
  {printf("%d ", arr[i]);  }}
```

Output:

```
Enter number of elements in the array :5
3
4
5
2
11
Enter the data you want to insert: 8
Output
3 8 4 5 2 11
Process returned 6 (0x6)   execution time : 12.557 s
Press any key to continue.
```

3) Find out the index of the 3rd largest value in an array and insert a value in that index.

⇒ Input elements, arr[ ] = {3, 4, 5, 2, 11}

Finding third largest number.

(a) When i = 1 = 3

if (array[1] > first Largest) [condition applies]
third Largest = second Largest
second Largest = first Largest
first Largest = arr[3]; i++

(b) i = 2 = 4

if (arr[2] > first Largest) [condition applies]
third largest = second Largest
second largest = first largest
first largest = arr[2]; i++

[3 is second largest,
4 is first largest]

(c) i = 3 = 5

if (arr[3] > first largest) [condition applies]
third largest = second largest
second largest = first largest
first largest = arr[3]. i++

[3 is third largest,
4 is second largest,
5 is first largest]

(d) i = 4 = 2

if (arr[4] > first largest)
else if (arr[4] > second largest)
else if (arr[4] > third largest)
no condition agrees so no change in the array. i++

[Condition doesn't apply]

(e) i = 5 = 11

if (arr[5] > first largest) [condition applies]
third largest = second largest
second largest = first largest
first largest = arr[5]

[4 is third largest,
5 is second largest,
11 is first largest]

3	4	5	2	11

Step-1:

$i = n = 5$, $i \geq$ third largest $= 2$

$\Rightarrow a[5+1] = a[5]$

$\Rightarrow a[6] = 11$; $i--$

3	4	5	2	11	11

Step-2:

$i = 4$, $i \geq 2$

$\Rightarrow a[4+1] = a[4]$

$\Rightarrow a[5] = 2$; $i--$

3	4	5	2	2	11

Step-3:

$i = 3$, $i \geq 2$

$\Rightarrow a[3+1] = a[3]$

$\Rightarrow a[4] = 5$, $i--$

3	4	5	5	2	11

Step-4:

$i = 2$, $i \geq 2$

$\Rightarrow a[2+1] = a[2]$

$\Rightarrow a[3] = 4$, $i--$

3	4	4	5	2	11

Step-5

$i = 1$, $i \geq 2$ [condition doesn't apply]

(loop ended)

$a[i] = a[2] = 8$

Step-6

low = 2, high = 2

$mid = \frac{low + high}{2} = \frac{2+2}{2} = 2$

if (arr[2] == target) [condition applies]

{: item is found

NOTRE DAME UNIVERSITY BANGLADESH

Data Structure

Report on Task- 04

Submitted to: HumayaraBinte Rashid

Submitted by: ShazidulAlam

Subject: Data Structure

Student ID: 0692220005101009

Batch: CSE-19

1. Implement a stack using the C language.

Output:

10 pushed into stack

20 pushed into stack

30 pushed into stack

30 Popped from stack

Top element is : 20

Elements present in stack : 20 10

```
#include <stdio.h>
int stack[4];
intMaxstk=4;
int Top=0;
int push(int value)
{if(value==Maxstk)
{printf("Overflow"); }
else{Top++;
stack[Top]=value;
return value; }}
int pop(int value)
{if (Top==0){
printf ("Underflow");    }
else{value=stack[Top];
    Top--;
return value; }}
void main()
{   printf("%d pushed into stack\n",push(10));
printf("%d pushed into stack\n",push(20));
printf("%d pushed into stack\n",push(30));
printf("%d popped from stack\n",pop (30));
printf("Top element is %d\n",stack[Top]);
printf("Element present in stack:");
for(int i= Top; i>0; i--){
printf("%d ",stack[i]);  }}
```

Output:

10 pushed into stack
20 pushed into stack
30 pushed into stack
30 popped from stack
Top element is 20
Element present in stack:20 10

1) Implement a stack

Step-1: When we push 10 in push function

=> if(10==(MaxStk=5)[False]

=> Top++

=> stack[Top]= 10

=> return 10

Step-2: Push 20 in function

=> if (20==(MaxStk=5))[False]

=> Top++

=> Stack[Top]=20

return 20.

Step-3: Push 30 in function.

=> if 30 == (MaxStk=5][False]

=> Top++

=> Stack[Top]=30

=> return 30.

Step-4: ~~Push 30~~ Pop 30 from ~~function~~ stack.

=> if(Top==0)[False]

=> int value = stack[Top]

=> Top--

return value.

Step-5:

~~We~~ Compiler will print element using reverse for loop from Top to 0.

2. Suppose your id is **06922300051010XY.** XY is a single variable which defines the last 2 digits of your id respectively. You are writing your id. Whenever you make a mistake, you can use control Z to undo the previous digit.

Sample Input:
Enter the last 2 digit of your id XY = 14

a	Enter: 0	j	Control Z
b	Enter: 6	k	Enter: 51
c	Enter: 9	l	Enter: 0
d	Enter: 9	m	Enter: 2
e	Control Z	n	Control Z
f	Enter: 22	o	Enter: 10
g	Enter: 300	p	Enter: 512
h	Enter: 0	q	Control Z
i	Enter: 0	r	Enter: 14

Sample Output: **0692230005101014**

```
#include<stdio.h>
int MAXSTK=16;
int stack[16];
int top=0;
int push(int item)
{  if(top==MAXSTK)
{   printf("The stack is already filled");
return 0;}
else{top=top+1;
stack[top]=item;
printf("Enter:%d\n",item);}}
int pop(int item)
{  if(top==-1){
printf("Stack is empty");
return 0;  }
else
{  top=top-1;
printf("Control Z\n",item);}}
int main()
{  intp,XY;
printf("Enter the last two digit of your id XY=");
scanf("%d",&XY);
   push(0);push(6);push(9);push(9);pop(9);push(22);push(300);push(0);push(0);pop(0);push(51);push(0);push(2);
   pop(2);push(10);push(512);pop(512);push(XY);for(p=1;p<=top;p++)
printf("%d",stack[p]);}
```

Output:

Enter the last two digit of your id XY=14

Enter:0
Enter:6
Enter:9
Enter:9
Control Z
Enter:22
Enter:300
Enter:0
Enter:0
Control Z
Enter:51
Enter:0
Enter:2
Control Z
Enter:10
Enter:512
Control Z
Enter:14

0692230005101014

2) Step-1: When we push 0 in Enter function, if (top == MAXSTK) X, top++, stack[top] = 0, print value.

Step-2: Push 6 in Enter function, if (top == MAXSTK) X, top++, stack[top] = 6, print the value.

Step-3: Execute the same function till we push 9 in Enter function.

Step-4: Implement control z function and pop the last value from the array.

Step-5: Repeat step-3 just push 22 and go until 0.

Step-6: Repeat step-4.

Step-7: Repeat step-5 just push 51 to 2.

Step-8: Repeat step-4.

Step-9: Repeat step-5 just push 10 to 512.

Step-10: Repeat step-4.

Step-11: Push 14 in Enter function.

Step-12: Print array values from index 1 to index top.

3. Find the numeric digit for any string.

Sample Input: hello527world81

Output: 5 2 7 8 1

```
#include <stdio.h>
int main() {
char input[100];
printf("Enter a string: ");
scanf("%s", input);
printf("Numeric digits in the string: ");
for (int i = 0; input[i] != '\0'; i++) {
if (input[i] >= '0' && input[i] <= '9') {
printf("%c ", input[i]); } }
printf("\n");
return 0;}
```

Output:

Enter a string: hello527world81

Numeric digits in the string: 5 2 7 8 1

3) The input is hello 527 world 81

When, input[i] = '\0' [True]

=> if (input[h] >= '0' && input[h] <= '9') [False] i++

When, input[i](e) != '\0' [True]

=> if (input[e] >= '0' && input[e] <= '9') [False] i++

input[i](l) != '\0' [True]

=> if (input[l] >= '0' && input[l] <= '9') [False] i++

input[i](l) != '\0' [True]

=> if (input[l] >= '0' && input[l] <= '9') [False] i++

input[i](o) != '\0' [True]

=> if (input[o] >= '0' && input[o] <= '9') [False] i++

input[i](5) != '\0' [True]

if (input[5] >= '0' && input[5] <= '9') [True]

print 5. i++

input[i](2) != '\0' [True]

=> if (input[2] >= '0' && input[2] <= '9') [True]

print 2. i++

input[i](7) != '\0' [True]

=> if (input[7] >= '0' && input[7] <= '9') [True]

print 7. i++

input[i](w) != '\0' [True]

=> if (input[w] >= '0' && input[w] <= '9') [False] i++

input[i](o) != '\0' [True]

=> if (input[o] >= '0' && input[o] <= '9') [False] i++

input[i](r) != '\0' [True]

=> if (input[r] >= '0' && input[o] <= '9') [False] i++

input[i](l) != '\0' [True]

=> if (input[l] >= '0' && input[l] <= '9') [False] i++

input[i](d) != '\0' [True]

=> if (input[d] >= '0' && input[d] <= '9') [False] i++

input[i](8) != '\0' [True]

=> if (input[8] >= '0' && input[8] <= '9') [True]

print 8. i++

input [i](1)! = '\0' [True]
=> if (input[1] >= '0' && input[1] <= '9')[True]
print 1. i++

NOTRE DAME UNIVERSITY BANGLADESH

Data Structure Report

Task- 05

Submitted to: HumayaraBinte Rashid

Submitted by: ShazidulAlam

Subject: Data Structure

Student ID: 0692220005101009

Batch: CSE-19

1. Find out the numeric digit from any string.
   Sample Input: hello527world81
   Output: 5 2 7 8 1

Input:

```
#include <stdio.h>
#include <ctype.h>
int main() {
char input[100];
printf("Enter a string: ");
scanf("%s", input);
printf("Numeric digits in the string: ");
for (int i = 0; input[i] != '\0'; i++) {
if (isdigit(input[i])) {
printf("%c ", input[i]); }    }
printf("\n");
return 0;}
```

Output:

```
Enter a string: hello527world81
Numeric digits in the string: 5 2 7 8 1

Process returned 0 (0x0)   execution time : 12.201 s
Press any key to continue.:
```

3) The input is hello 527 world & 1

When, input[i] = '\0' [True]

=> if (input[h] >= '0' && input[h] <= '9') [False] i++

When, input[i](e) != '\0' [True]

=> if (input[e] >= '0' && input[e] <= '9') [False] i++

input[i](l) != '\0' [True]

=> if (input[l] >= '0' && input[l] <= '9') [False] i++

input[i](l) != '\0' [True]

=> if (input[l] >= '0' && input[l] <= '9') [False] i++

input[i](o) != '\0' [True]

=> if (input[o] >= '0' && input[o] <= '9') [False] i++

input[i](5) != '\0' [True]

if (input[5] >= '0' && input[5] <= '9') [True]

print 5. i++

input[i](2) != '\0' [True]

=> if (input[2] >= '0' && input[2] <= '9') [True]

print 2. i++

input[i](7) != '\0' [True]

=> if (input[7] >= '0' && input[7] <= '9') [True]

print 7. i++

input[i](w) != '\0' [True]

=> if (input[w] >= '0' && input[w] <= '9') [False] i++

input[i](o) != '\0' [True]

=> if (input[o] >= '0' && input[o] <= '9') [False] i++

input[i](r) != '\0' [True]

=> if (input[r] > '0' && input[o] <= '9') [False] i++

input[i](l) != '\0' [True]

=> if (input[l] >= '0' && input[l] <= '9') [False] i++

input[i](d) != '\0' [True]

=> if (input[d] >= '0' && input[d] <= '9') [False] i++

input[i](8) != '\0' [True]

=> if (input[8] >= '0' && input[8] <= '9') [True]

print 8. i++

2. Convert any character into an integer.
   Sample input: '8'
   Output: 8

Input:

```
#include <stdio.h>
intcharToInt(char c) {
return c - '0';}//To identify characters.
int main() {
charinputChar;
printf("Enter a character representing a digit: ");
scanf(" %c", &inputChar);
int digit = charToInt(inputChar);
printf("Output: %d\n", digit);
return 0;}
```

Output:

```
Enter a character representing a digit: 8
Output: 8

Process returned 0 (0x0)   execution time : 10.243 s
Press any key to continue.
```

2) ① Declaring an array of characters 'arr' with a size of 100.

② It prompts the user to enter a number as a string with 'printf".

③ You enter "8" as input.

④ The "scanf" function reads the input and stores it in the array.

⑤ The "atoi" function is used to convert the string "1" to an integer and the result is stored in the 'converted number" variable.

⑥ Finally, it prints the converted number using 'printf' and you will see "converted number is 1" as the output.

3. Take a user input in files and show the output in the terminal for an array.

Sample input:

5

11 4 36 78 29

Output:

A[0]=11 A[1]=4 A[2]=36 A[3]=78 A[4]=29

Input:

```
#include <stdio.h>
int main() {
intarr[100];
   FILE *inputFile = fopen("id1.txt", "r");
int size;
fscanf(inputFile,"%d", &size);
printf("Array elements:\n");
for (int i = 0; i < size; i++) {
fscanf(inputFile, "%d ", &arr[i]);   }
printf("Array elements:\n");
for (int i = 0; i < size; i++) {
printf("%d ", arr[i]);
   }
printf("\n");
return 0;}
```

Output:

```
Array elements:
Array elements:
11 4 36 78 29

Process returned 0 (0x0)   execution time : 0.028 s
Press any key to continue.
```

3) By including the necessary header files and defining the 'main' function.

1) The program prompts the user to specify the size of an array (n).

2) It creates an integer array 'arr' of size n to hold user-provided values.

3) Users are instructed to input n integers separated by spaces.

4) The program reads and stores these integers in the array.

5) It then proceeds to display the array elements.

6) The output format for each element is "A[index]=value".

7) The loop iterates through the array, printing each element with its corresponding index.

8) It closes the file using 'fclose'.

4. Take a string and convert it to a decimal number.
   Sample input: ha2c3s4
   Output: 200+30+4=234

Input:

```
#include <stdio.h>
#include <ctype.h>
int main() {
intdecimal_number = 0;
char string[100];
printf("Enter the string: ");
scanf("%s", string);
for (int i = 0; string[i] != '\0'; i++) {
if (isdigit(string[i])) {
decimal_number = (decimal_number * 10) + (string[i] - '0');
        }
    }
printf("The decimal number is %d\n", decimal_number);
return 0;
}
```

Output:

```
Enter the string: ha2c3s4
The decimal number is 234

Process returned 0 (0x0)   execution time : 21.650 s
Press any key to continue.
```

// The input is ha2c34

string [i] = h

for (int i = 0; string [i] ! = '0'; i++) [True]

=> if (is digit (string [i])) [False] i++

string [i] = a

for (int i = 0; string [i] ! = '0'; i++) [True]

=> if (is digit (stairy [i])) [False] i++

string [i] = 2

for (int i = 0; string [i] ! = '0'; i++) [True]

=> if (is digit (string [i])) [True]

decimal number (0) = (decimal number * 10) [0] + (string [2] - '0');

∴ decimal number = 0 + 2 = 2 . i++

string [i] = c

for (int i = 0; string [i] ! = '0'; i++) [True]

=> if (is digit (string [i])) [False] . i++

string [i] = 3

for (int i = 0; string [i] ! = '0'; i++) [True]

=> if (is digit (string [i])) [True]

decimal_number (2) = (decimal_number * 16) [20] +

(string [3] - '0');

∴ decimal number = 20 + 3 = 23

string [i] = 5

for (int i = 0; string [i] != '0'; i++) [True]

=> if (is digit (string [i])) [False] . i++

string [i] = 4

for (int i = 0, string [i] ! = '\0'; i++) [True]

=> if (is digit (string [i])) [True]

decimal_number (23) = (decimal_number * 10) [230] +

(string [4] - '0'; ∴ decimal - number = 230 + 4 = 234.

i++

* String [i] = \0

for (int i = 0; string [i] ! = 'p'; i++) [False]

[loop ended]

* print decimal - number.

5. Take a series of numbers as input in only one string using space to differentiate one number from another and convert them into numbers. Then find out the summation of total numbers.

Sample input: 10 20 5 15

Sample output:

Number: 10
Number: 20
Number: 5
Number: 15
Sum: 50

Input:

```
#include <stdio.h>
#include <stdlib.h>
int main() {
char input[100];
printf("Enter a series of numbers separated by spaces: ");
fgets(input, sizeof(input), stdin);
int sum = 0;
int number;
intnumCount = 0;
printf("Output:\n");
for (int i = 0; input[i] != '\0'; i++) {
if (input[i] >= '0' && input[i] <= '9') {
number = 0;
while (input[i] >= '0' && input[i] <= '9') {
number = number * 10 + (input[i] - '0');
i++;          }
printf("Number: %d\n", number);
sum += number;
numCount++;       }   }
printf("Sum: %d\n", sum);
return 0;}
```

Output:

Enter a series of numbers separated by spaces: 10 20 5 15
Output:
Number: 10
Number: 20
Number: 5
Number: 15
Sum: 50

5)

1) The program initializes a character array 'input' to store the user input.

2) It prompts the user to enter a series of numbers separated by spaces and reads the input using 'fgets'.

3) An integer 'sum' is initialized to zero to store the summation of the numbers.

4) An integer 'number' is used to temporarily store the extracted numbers.

5) An integer 'numCount' is set to zero to keep track of the number of valid numbers extracted.

6) The program processes the input string character by character, checking if each character is a digit (0-9).

7) When a digit is encountered, the program extracts and converts it into an integer by looping through the consecutive digits in the input.

8) For each valid number extracted, it prints "Number: [number]" and updates the sum and numCount. After processing the entire input, it displays "Sum: [sum]" as the total sum of the extracted numbers.

NOTRE DAME UNIVERSITY BANGLADESH

Data Structure Report

Post Order (Sep 10)

Submitted to: HumayaraBinte Rashid

Submitted by: ShazidulAlam

Subject: Data Structure

Student ID: 0692220005101009

Batch: CSE-19

Post Order

```
#include<stdio.h>
int n=15;
char T[]={'\0','D','B','F','A','C','E','G','\0','\0','\0','\0','\0','\0','H'};
intlc(int i)
{
   if (T[i]!='\0' && (2*i)<=n)
      return (2*i);
   return -1;
}
intrc(int i)
{
   if (T[i]!='\0' && ((2*i)+1)<=n)
      return (2*i)+1;
   return -1;
}
void postorder (int i)
{
   if (i>0 && T[i]!='\0')
   {postorder(lc(i));
   postorder(rc(i));
   printf("%c", T[i]);}
}
int main()
{
   int a ;
   printf("please enter the root index :\n");
   scanf("%d",&a);
   postorder(a);
   return 0;
}
```

NOTRE DAME UNIVERSITY BANGLADESH

Data Structure Report

Task- 07

Submitted to: HumayaraBinte Rashid

Submitted by: ShazidulAlam

Subject: Data Structure

Student ID: 069222005101009

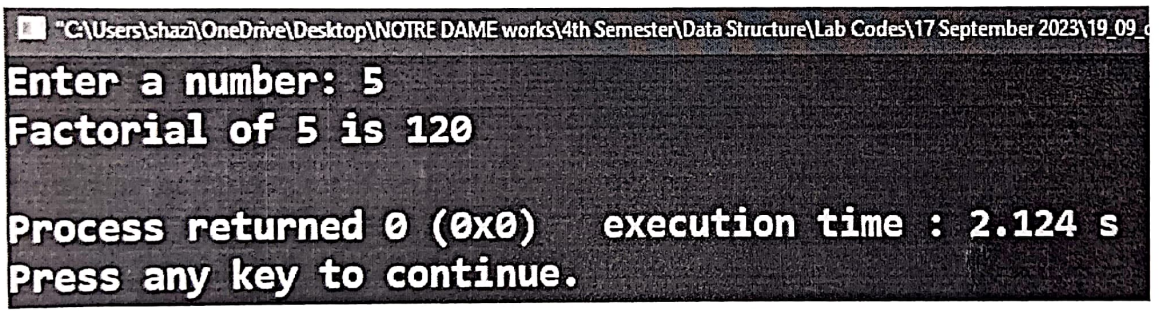
Batch: CSE-19

1(a).Write a code to find the factorial of N using a)recursion.
Sample Input: 5
Output: 120

Soution:

```
#include <stdio.h>
int factorial(int n) {
if (n == 0 || n == 1) {
return 1;
   } else {
return n * factorial(n - 1);  }
}
int main() {  int n;
printf("Enter a number: ");
scanf("%d", &n);
if (n < 0) {
printf("Factorial is not defined for negative numbers.\n");
   } else {
int result = factorial(n);
printf("Factorial of %d is %d\n", n, result);  }
return 0;}
```

Output:



1(a)

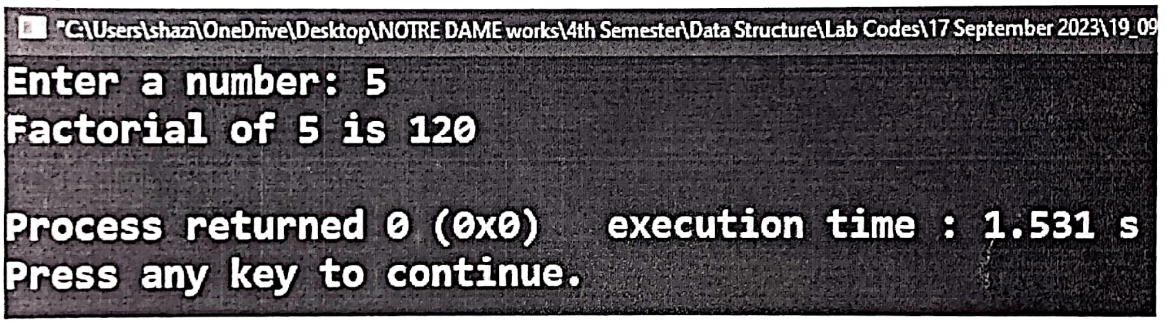
1) The code defines a recursive function named 'factorial' to calculate the factorial of a given number 'n'.
2) Inside the 'factorial' function, it checks if 'n' is equal to 0 or 1, and if true, returns 1, serving as the base case for recursion.
3) If 'n' is greatere than 1, it recursively calculates in *factorial (n-1)' to find the factorial.
4) In the 'main' function, it takes an integere 'n' as input from the user.
5) It checks if 'n' is negative and handles this case by displays a message that factorial is not defined for negative numbers.
6) If 'n' is non-negative, it calculates the factorial using the 'factorial' function.
7) The calculated factorial result is displayed in the foremat "Factorial of N is X" where N is the input number and X is the factorial
8) The code is executed and correctly computes and displays the factorial for the provided input. For example, for 'N = 5,' it will output 'Factorial of 5 is 120'.

1(b).Write a code to find the factorial of N using b)using for loop.
Sample Input: 5
Output: 120

Solution:

```
#include <stdio.h>
int main() {
int N;
printf("Enter a number: ");
scanf("%d", &N);
if (N < 0) {
printf("Factorial is not defined for negative numbers.\n");
   } else {
int result = 1;
for (int i = 1; i <= N; i++) {
result =result*i;
      }
printf("Factorial of %d is %d\n", N, result);
   }
return 0;}
```

Output:



1.(b)

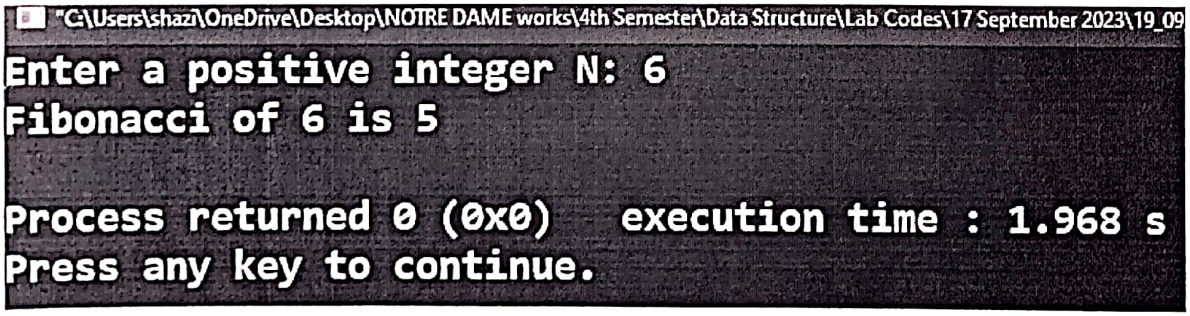
1) The code begins by taking an integer 'N' as input from the user.
2) It checks if 'N' is negative and if it is, it displays a message indicating that factorial is not defined for negative.
3) If 'N' is non-negative, it initializes an integer variable 'result' to 1. This variable will store the factorial.
4) The code then uses a for loop to calculate the factorial. It iterates from 'i=1' to 'i<=N'.
5) Inside the loop, it updates the 'result' by multiplying it with 'i' in each iteration.
6) After the loop completes, it displays the calculated factorial in the format "Factorial of N is X," where N is the input number and X is the factorial.
7) The code is executed and correctly computes and displays the factorial for the provided input.
8) For the given sample input '5', the code will output 'Factorial of 5 is 120'.

2. Write a code to find the Nth number of fibonacci series with recursion.
Sample input: 6
Output: 5
Bonus: print till Nth number.
Series Output: 0 1 1 2 3 5

Solution:

```
#include <stdio.h>
intfibonacci(int n) {
if (n == 1) {
return 0;
   }
else if (n==2){
return 1;
   }
else {
returnfibonacci(n - 1) + fibonacci(n - 2);    }
}
int main() {
int N;
printf("Enter a positive integer N: ");
scanf("%d", &N);
if (N < 0) {
printf("Invalid input. N should be a positive integer.\n");    }
else {
int result = fibonacci(N);
printf("Fibonacci of %d is %d\n", N, result);
   }
return 0;}
```

Output:



(Task-7)

2)

1) The code defines a recursive function named 'fibonacci' to calculate the Nth number in the Fibonacci series.

2) Inside the 'fibonacci' function, it checks if 'n' is equal to 1 or 2. If 'n' is 1, it returns 0. and if 'n' is 2, it returns 1. These are the base cases of the Fibonacci series.

3) For 'n' greater than 2, the 'fibonacci' function recursively calculates 'fibonacci(n-1)+ fibonacci(n-2)' to find the Nth number in the Fibonacci series.

4) In the 'main' function, it takes a positive integer 'N' as input from the user.

5) It checks if 'N' is negative and if it is, it displays a message indicating that the input is invalid, as 'N' should be a positive integer.

6) If 'N' is valid and non-negative, it calls the 'fibonacci' function to calculate the Nth Fibonacci number.

7) The calculated Fibonacci result is displayed in the format "Fibonacci of N is X," where N is the input number and X is the Nth Fibonacci number.

NOTRE DAME UNIVERSITY BANGLADESH

Data Structure Report

Adjacecncy Matrix

Submitted to: HumayaraBinte Rashid

Submitted by: ShazidulAlam

Subject: Data Structure

Student ID: 069222005101009

Batch: CSE-19

Adjacency Matrix:

```
#include<stdio.h>
main()
{
inti,j,p,q,n,e;
printf ("Enter the number of vertex : ");
scanf("%d",&n);
printf ("Enter the number of edge : ");
scanf("%d",&e);
int a[n+1][n+1];
for(i=1; i<=n; i++)
{
for(j=1; j<=n; j++)
{
a[i][j]=0;
}
}
for(i=1; i<=n; i++)
{
//printf("%d:",i);
for (j=1; j<=n; j++)
{printf ("%d ",a[i][j]);
}printf("\n");
}printf("Enter the edges:\n");
for(int k=1; k<=e; k++)
{scanf("%d %d",&p,&q);
a[p][q]=1;
//a[q][p]=1;
}for(i=1; i<=n; i++)
{
//printf("%d:",i);
for (j=1; j<=n; j++)
{
printf ("%d ",a[i][j]);
}
printf("\n");
}
return 0;}
```

(Adjacency Matrix)

1) The code initiates by prompting the user to input the number of vertices (n) and edges (e) for the graph.

2) It declares a 2D integer array 'a' of size (n+1) x (n+1) to serve as the adjacency matrix, initialized with zeros.

3) The code proceeds to display the initial adjacency matrix with all element set to 0. This initial matrix represents no edges between vertices.

4) The user is then asked to input the edges of the graph by specifying the source and destination vertices. This input stage allows for the definition of the graph's connectivity.

5) In a loop that iterates for the number of edges (e), the code reads and records the edges by setting the appropriate elements in the adjacency matrix to 1. This action signifies the presence of an edge between the corresponding vertices.

6) The code efficiently updates the adjacency matrix based on the edges entered by the user, gradually constructing the graph's connectivity.

7) Once all edges have been processed and the adjacency matrix is fully updated, it proceeds to display the matrix. This final

adjacency matrix illustrates the connections between the vertices, with '1' at position 'a[i][j]' indicating an edge between vertex i and vertex j.

8) The program ensures that it properly handles the graph's adjacency matrix and provides a visual representation of the graph's connectivity.

9) It employs a straightforward method to allow users to define edges and visualize the graph's structure.

10) The code follows a structured approach to create an adjacency matrix and shows the connections within the graph.

11) After displaying the updated adjacency matrix, the code successfully conveys the relationships between vertices based on the user's input.

12) The code execution concludes, having achieved the task of generating and visualizing an adjacency matrix for a graph, representing its connectivity.