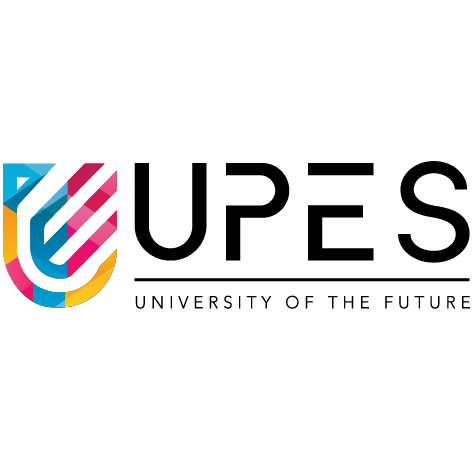
**School of Computer Science**

**UNIVERSITY OF PETROLEUM AND ENERGY STUDIES**

**DEHRADUN, UTTARAKHAND**



**Data Structures Lab**

**(CSEG 1111)**

**Lab File**

**(2021-2022)**

**for**

**2nd Semester**

**Submitted By:**

*Mridul Vasudeva*

B. Tech. CSE [2nd Semester]

*R2142210502*

*Batch - 1*

**Submitted To:**

Mr. Pushpendra K Rajput

Assistant Professor-SS

Cybernetics Cluster,

School of Computer Science

Experiment-01

**Title:1-Array &Structure**

**Objective:**To apply the concept of array, structure and experiment on nested array and array of structures.

Problem - 1.1L: Find sum of all array elements using recursion.

**Solution Approach :** The program inputs number of elements for the array from the user, creates an array of particular size, takes input from the user and then goes on to add the elements of the array.

A recursive function calls itself again and again to add up the elements and breaks once it reaches to the end of array.

**Source Code:**

// 17th January , 2022

// Lab Activity - 1 , Question 1.

// Mridul Vasudeva

// A C program used to find the sum of array using recursion.

#include <stdio.h>

void ArrayInput (int array[] , int m) {

for (int i=0; i<m; i++) {

printf("Enter %d element: ",i+1);

scanf("%d",&array[i]);

}

}

// Function using recursion to find sum of elements of array.

int SumofArray (int array[] , int x , int m) {

int sum = 0;

// Base condition used to terminate the iterations recursion will be performing.

if (x == m)

return sum;

else

// Here sum of elements of array is calculated.

return sum = array[x] + SumofArray(array , x+1 , m);

}

void main () {

int m;

printf("Enter size of array: ");

scanf("%d",&m);

int array[m];

// Main is calling a function used to take input of array elements.

ArrayInput(array , m);

// Here s is storing sum of array...

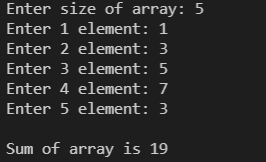
int s = SumofArray(array , 0 , m);

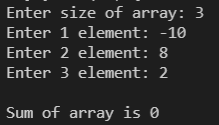
// And printing it as an output here.

printf("\nSum of array is %d\n",s);

}

**Sample Output:**





**Title:1-Array &Structure**

**Objective:**To apply the concept of array, structure and experiment on nested array and array of structures.

Problem - 1.2L: Create an array ‘a1’with ‘n’ elements. Insert an element in ith position of ‘a1’ and also delete an element from jth position of ‘a1’

**Source Code:**

// 17th January , 2022

// Lab Activity - 1 , Question 2.

// Mridul Vasudeva

// A C program used to insert and delete elements of array using switch case statements.

#include <stdio.h>

#include <stdlib.h>

// A function InsertElement is called when user choses choice 1.

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

int index , insert\_element;

printf("\nEnter index to insert an element: ");

scanf("%d",&index);

printf("Enter element to insert on %d position: ",index);

scanf("%d",&insert\_element);

n = n+1;

// Loop used to insert an element and shift other elements.

for (int j=n; j>=index; j--) {

\*(arr+j+1) = \*(arr+j);

}

arr[index] = insert\_element;

// Loop used to print the array.

for (int k=0; k<n; k++) {

printf("%d ",\*(arr+k));

}

}

// A function DeleteElement is called when user choses choice 2.

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

int index\_delete;

printf("\n\nEnter index of element to be deleted: ");

scanf("%d",&index\_delete);

// Conditions used to check wheter a element can be deleted or not.

if (index\_delete >= n+1) {

printf("\nDeletion not possible.");

} else {

// Loop used to delete an element and shift other element to the left.

for (int x=index\_delete-1; x<n-1; x++) {

arr[x] = arr[x+1];

}

}

printf("\nThe resultant array is:\n");

// Loop used to print the resultant array.

for (int x=0; x<n-1; x++) {

printf("%d ",\*(arr+x));

}

}

void main () {

int n;

printf("Enter length of array: ");

scanf("%d",&n);

int arr[100];

// Loop used to take input of array elements.

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

printf("Enter %d element: ",i+1);

scanf("%d",arr+i);

}

// Menu

printf("\n\t\tPress:");

printf("\n\t1 to insert an element.");

printf("\n\t2 to delete an element.");

printf("\n\t3 to Exit");

printf("\n\tEnter your choice: ");

int ch;

scanf("%d",&ch);

// Using switch-case to provide a menu to the user.

switch (ch) {

case 1: {

InsertElement(arr , n);

// Break statement used to stop the flow of program after executing choice 1.

break;

}

case 2: {

DeleteElement(arr , n);

// Break statement used to stop the flow of program after executing choice 2.

break;

}

case 3: {

// Exit function used to stop the program as per users

exit(0);

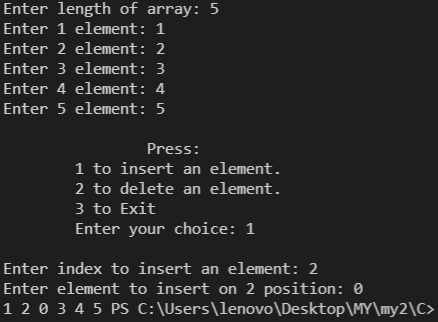
// Break statement used to stop the flow of program after executing choice 3.

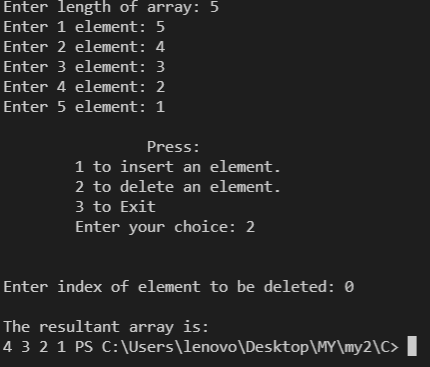
break;

}

}}

**Sample Output:**





**Title:1-Array &Structure**

**Objective:**To apply the concept of array, structure and experiment on nested array and array of structures.

Problem - 1.3L: Convert a Uppercase string to lowercase using for loop.

**Source Code:**

// 17th January , 2022

// Lab Activity - 1 , Question 3.

// Mridul Vasudeva

// A C program to convert Uppercase to Lowercase.

#include <stdio.h>

void main () {

// Initializing a string of 50 characters.

char str[50];

// Taking input of string.

printf("Enter string in Uppercase: ");

gets(str);

// Using loop to convert Uppercase string to lowercase string.

for (int i=0; str[i]!='\0'; i++) {

if (str[i] >= 'A' && str[i] <= 'Z')

str[i] = str[i] + 32;

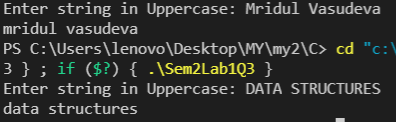
}

// Printing the string converted into lowercase.

puts(str);

}

**Sample Output:**

****

**Title:1-Array &Structure**

**Objective:**To apply the concept of array, structure and experiment on nested array and array of structures.

Problem - 1.4L: Find sum of rows and columns of matrix given order(row x column).

**Source Code:**

// 17th January, 2022

// Lab Activity - 1, Question – 4

// A C program used to find sum of rows and columns of a given 2D array.

#include <stdio.h>

void main() {

int m , n;

printf("Enter size of matrix [m] [n]: ");

scanf("%d %d",&m,&n);

// A 2D matrix of size mxn is created as per users requirement.

int array[m][n];

// Loops take input of array elements.

for (int i=0; i<m; i++) {

for (int j=0; j<n; j++) {

printf("Enter [%d][%d] element: ",i,j);

scanf("%d",&array[i][j]);

}

}

printf("\n");

// Loops print the matrix.

for (int i=0; i<m; i++) {

for (int j=0; j<n; j++) {

printf("%d ",array[i][j]);

}

printf("\n");

}

int sumrow , sumcol;

// Finding sum of row.

for (int i=0; i<m; i++) {

sumrow = 0;

for (int j=0; j<n; j++) {

sumrow = sumrow + array[i][j];

}

printf("Sum of row %d: %d\n",i+1,sumrow);

}

printf("\n");

// Finding sum of column.

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

sumcol = 0;

for (int j=0; j<m; j++) {

sumcol = sumcol + array[j][i];

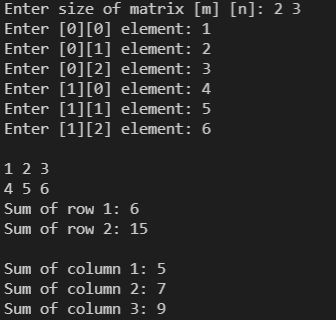
}

printf("Sum of column %d: %d\n",i+1,sumcol);

}

}

**Sample Output:**

****

**Title:1-Array &Structure**

**Objective:**To apply the concept of array, structure and experiment on nested array and array of structures.

Problem - 1.5L: Find product of two matrices.

**Source Code:**

// 17th January , 2022

// Lab Activity - 1 , Question 3.

// Mridul Vasudeva

// A C program to Multiply two matrices.

#include <stdio.h>

void main() {

int m , n;

printf("Enter size of matrix [m] [n]: ");

scanf("%d %d",&m,&n);

// A 2D matrix of size mxn is created as per users requirement.

// Since matrix multiplication is possible between matrices of same size.

int array1[m][n] , array2[m][n] , array3[m][n];

printf("\nInput elements for 1st array:\n");

for (int i=0; i<m; i++) {

for (int j=0; j<n; j++) {

printf("Enter [%d][%d] element: ",i,j);

scanf("%d",&array1[i][j]);

}

}

printf("\n");

printf("Input elements for 2nd array:\n");

for (int i=0; i<m; i++) {

for (int j=0; j<n; j++) {

printf("Enter [%d][%d] element: ",i,j);

scanf("%d",&array2[i][j]);

}

}

int arraymul;

for (int i=0; i<m; i++) {

for (int j=0; j<n; j++) {

arraymul = 0;

for (int x=0; x<n; x++) {

arraymul += array1[i][x]\*array2[x][j];

}

array3[i][j] = arraymul;

}

}

printf("\nMultiplication of two matrices is:\n");

for (int i=0; i<m; i++) {

for (int j=0; j<n; j++) {

printf("%d ",array3[i][j]);

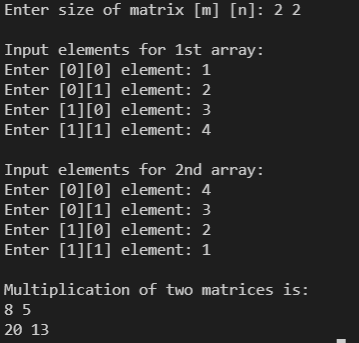
}

printf("\n");

}

}

**Sample Output:**

****

**Title:1-Array &Structure**

**Objective:**To apply the concept of array, structure and experiment on nested array and array of structures.

Problem - 1.6L: Store ‘n’ numbers of integers in an array. Conduct a linear search for a given number and report success or failure in form of a suitable message.

**Source Code:**

// 17th January , 2022

// Lab Activity - 1 , Question 5.

// Mridul Vasudeva

// A C program used to check if a given element is present inside array or not.

#include <stdio.h>

int SearchElement(int array[] , int n , int key) {

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

if(array[i] == key) {

// Returns 1 if element is found.

return 1;

}

}

// Returns 0 if element not found.

return 0;

}

void main() {

int n , key;

printf("Enter number of elements: ");

scanf("%d",&n);

// Array created which can store both integers and real numbers.

int array[n];

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

printf("Enter %d element: ",i+1);

scanf("%d",&array[i]);

}

printf("\nEnter element to search for: ");

scanf("%d",&key);

int x = SearchElement(array , n , key);

if(x == 1) {

printf("\nElement successfully found!\n");

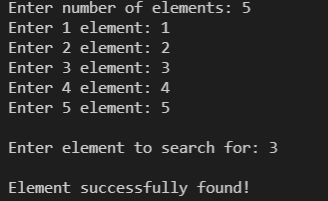
} else {

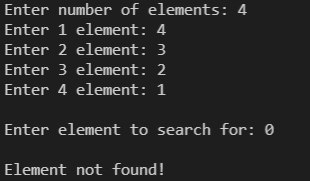
printf("\nElement not found!\n");

}

}

**Sample Output:**

****

****

**Practice Activity - 1**

**Title:1-Array &Structure**

**Objective:**To apply the concept of array, structure and experiment on nested array and array of structures.

Problem - 1.1P: Find the transpose and inverse of a matrix.

**Source Code:**

// 17th January , 2022

// Practice Activity - 1 , Question 1.

// Mridul Vasudeva

// A C program to find transpose and inverse of a matrix.

#include <stdio.h>

void main() {

int m , n;

printf("Enter size of matrix [m]x[n]: ");

scanf("%d %d",&m,&n);

float array[m][n];

printf("\n");

for (int i=0; i<m; i++) {

for (int j=0; j<n; j++) {

printf("Enter [%d][%d] element: ",i,j);

scanf("%f",&array[i][j]);

}

}

printf("\nOriginal Matrix:\n");

for (int i=0; i<m; i++) {

for (int j=0; j<n; j++) {

printf("%g ",array[i][j]);

}

printf("\n");

}

// Calculates transpose of matrix

printf("\nTranspose of matrix:\n");

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

for (int j=0; j<m; j++) {

printf("%g ",array[j][i]);

}

printf("\n");

}

// Calculates inverse of matrix

printf("\nInverse of the matrix:\n");

if(m == 2 && n == 2) {

float temp = array[0][0];

array[0][0] = array[1][1];

array[1][1] = temp;

array[0][1] = array[0][1]\*(-1);

array[1][0] = array[1][0]\*(-1);

float det2x2 = array[0][0]\*array[1][1] - array[0][1]\*array[1][0];

for (int i=0; i<2; i++) {

for (int j=0; j<2; j++) {

printf("%g ",(1/det2x2)\*array[i][j]);

}

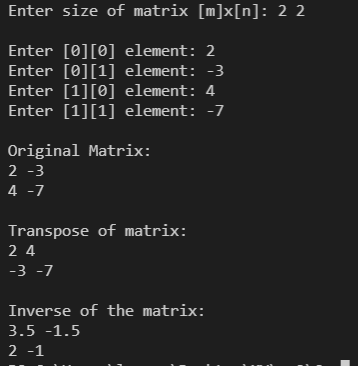
printf("\n");

}

}

}

**Sample Output:**

****

**Title:1-Array &Structure**

**Objective:**To apply the concept of array, structure and experiment on nested array and array of structures.

Problem - 1.2P: Find if a given matrix of size (mxn) is a sparse matrix or not.

**Source Code:**

// 17th January , 2022

// Practice Activity - 1 , Question 2.

// Mridul Vasudeva

// A C program to find if a given matrix is a sparse matrix or not.

#include <stdio.h>

// Function calculates number of zeros in the given matrix.

int FindSparse(int \*array , int m , int n) {

int count = 0;

for (int i=0; i<m; i++) {

for (int j=0; j<n; j++) {

if (\*(array+i+j) == 0) {

count++;

}

}

}

return count;

}

void main() {

int m , n;

printf("Enter size of matrix[m] [n]: ");

scanf("%d %d",&m,&n);

int array[m][n];

for (int i=0; i<m; i++) {

for (int j=0; j<n; j++) {

printf("Enter [%d][%d] element: ",i,j);

scanf("%d",&array[i][j]);

}

}

int ele = m\*n;

int c = FindSparse((int \*)array , m , n);

// Compares the number of zeros with total number of elements.

if(c > (ele/2)) {

printf("\nIt is a Sparse Matrix\n");

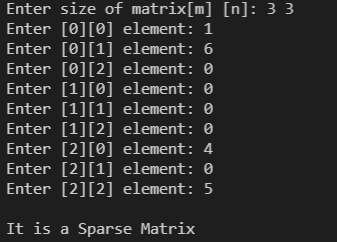
} else {

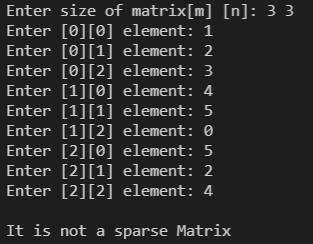
printf("\nIt is not a sparse Matrix\n");

}

}

**Sample Output:**

****

****

**Title:1-Array &Structure**

**Objective:**To apply the concept of array, structure and experiment on nested array and array of structures.

Problem - 1.3P: Store ‘n’ numbers in an array in ascending or descending order. Conduct a binary search for a given number and report success or failure in form of a message.

**Source Code:**

// 17th January , 2022

// Practice Activity - 1 , Question 3.

// Mridul Vasudeva

// Binary searching.

#include <stdio.h>

// Function searches for element and returns its index if found otherwise function returns -1 if element does not exist.

int BinarySearch (int array[] , int n , int key) {

int start = 0;

int end = n - 1;

while (start <= end) {

int mid = (start + end)/2;

if(array[mid] > key) {

end = mid - 1;

} else if(array[mid] < key) {

start = mid + 1;

} else {

return mid;

}

}

return -1;

}

void main() {

int n;

printf("Enter number of elements: ");

scanf("%d",&n);

int array[n];

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

printf("Enter element: ");

scanf("%d",&array[i]);

}

int key;

printf("\nEnter element to be found: ");

scanf("%d",&key);

int x = BinarySearch(array , n , key);

// Compares the value BinarySearch returns.

if(x == -1) {

printf("\nElement not found\n");

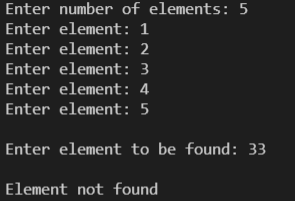
} else {

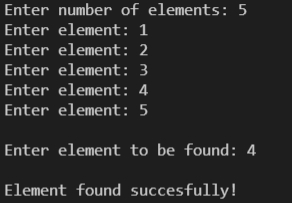
printf("\nElement found succesfully!\n");

}

}

**Source Code:**

****

****

**Title:1-Array &Structure**

**Objective:**To apply the concept of array, structure and experiment on nested array and array of structures.

Problem - 1.4P: Find out the largest and the smallest element from the array.

**Source Code:**

// 17th January , 2022

// Practice Activity - 1 , Question 4.

// Mridul Vasudeva

// A C program to find the largest and smallest number in an array.

#include <stdio.h>

// Returns the largest element from the array.

int FindLargest(int array[] , int n) {

int max = array[0];

for (int i=1; i<n; i++) {

if(array[i] > max) {

max = array[i];

}

}

return max;

}

// Returns the smallest element from the array.

int FindSmallest(int array[] , int n) {

int min = array[0];

for(int i=1; i<n; i++) {

if(array[i] < min) {

min = array[i];

}

}

return min;

}

void main() {

int n;

printf("Enter number of elements: ");

scanf("%d",&n);

int array[n];

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

printf("Enter elements: ");

scanf("%d",&array[i]);

}

int m = FindLargest(array , n);

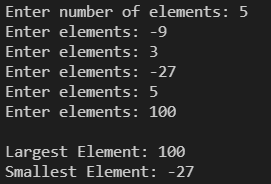
printf("\nLargest Element: %d",m);

int s = FindSmallest(array , n);

printf("\nSmallest Element: %d\n",s);

}

**Sample Output:**

****

**Title:1-Array &Structure**

**Objective:**To apply the concept of array, structure and experiment on nested array and array of structures.

Problem - 1.5P: Create an array ‘a1’ copy all the elements of array ‘a1’ to another array ‘a2’ and display elements of both the arrays using pointer.

**Source Code:**

// 17th January , 2022

// Practice Activity - 1 , Question 5.

// Mridul Vasudeva

#include <stdio.h>

void ArrayToArray(int\* array1 , int\* array2 , int n) {

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

\*(array2+i) = \*(array1+i);

}

printf("\nElements of Array 1:\n");

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

printf("%d ",\*(array1+i));

}

printf("\nElements of Array 2:\n");

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

printf("%d ",\*(array2+i));

}

}

void main() {

int n;

printf("Enter length of array: ");

scanf("%d",&n);

int array1[n];

int array2[n];

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

printf("Enter element: ");

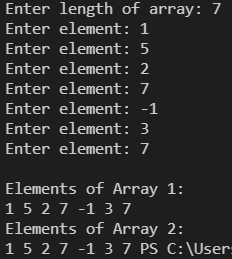
scanf("%d",&array1[i]);

}

ArrayToArray(array1 , array2 , n);

}

**Sample Output:**

****

**Title:1-Array &Structure**

**Objective:**To apply the concept of array, structure and experiment on nested array and array of structures.

Problem - 1.6P: Find number of Alphabets, Digits and special character in an integer.

**Source Code:**

// 17th January , 2022

// Practice Activity - 1 , Question 4.

// Mridul Vasudeva

// A C program to count number of alphabets , digits and speacial characters.

#include <stdio.h>

void main() {

char str[50];

printf("Enter a string: ");

gets(str);

printf("\nYour string is: %s",str);

int Alphcount = 0 , Digcount = 0 , Spccount = 0;

for (int i=0; str[i]!='\0'; i++) {

if(str[i] >= 'a' && str[i] <= 'z' || str[i] >= 'A' && str[i] <= 'Z') {

Alphcount++;

} else if(str[i] >= '0' && str[i] <= '9') {

Digcount++;

} else {

Spccount++;

}

}

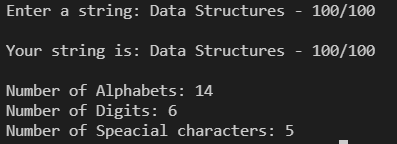
printf("\n\nNumber of Alphabets: %d",Alphcount);

printf("\nNumber of Digits: %d",Digcount);

printf("\nNumber of Speacial characters: %d\n",Spccount);

}

**Sample Output:**

****

**Title:1-Array &Structure**

**Objective:**To apply the concept of array, structure and experiment on nested array and array of structures.

Problem - 1.7P: Count the number of words in a string.

**Source Code:**

// 17th January , 2022

// Practice Activity - 1.

// Mridul Vasudeva

// A C program to count number of words in a string.

#include <stdio.h>

void main() {

char str[100];

printf("Enter a string: ");

gets(str);

printf("\nYour string is: %s",str);

int WordCount = 0;

for (int i=0; str[i]!='\0'; i++) {

if(str[i] == ' ' && str[i+1] != ' ') {

WordCount++;

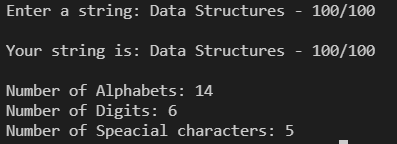
}

}

printf("\nNumber of words: %d\n",++WordCount);

}

**Sample Output:**

****

**Title:1-Array &Structure**

**Objective:**To apply the concept of array, structure and experiment on nested array and array of structures.

Problem - 1.8P: Add, Subtract and multiply two complex numbers using structure to a function as argument.

**Source Code:**

**// 17th January , 2022**

// Practice Activity - 1.

// Mridul Vasudeva

// A C program to Add, Subtract and multiply two complex numbers.

#include <stdio.h>

struct complex {

int real;

int img;

};

struct complex num1 , num2;

void AddComplex (struct complex num1 , struct complex num2) {

int AddReal = (num1.real + num2.real);

int Addimg = (num1.img + num2.img);

printf("\nAddition of two complex numbers is: %d + %di ",AddReal,Addimg);

}

void SubtractComplex (struct complex num1 , struct complex num2) {

int SubReal = (num1.real - num2.real);

int Subimg = (num1.img - num2.img);

printf("\nSubtraction of two complex numbers is: %d + %di ",SubReal,Subimg);

}

void MultiplyComplex (struct complex num1 , struct complex num2) {

int mulreal = (num1.real\*num2.real);

int mulrimg = (num1.real\*num2.img);

int mulimgrl = (num2.real\*num2.img);

int mulimgi = -(num1.img\*num2.img);

int addimg = mulimgrl + mulrimg;

int addimgrl = mulreal + mulimgi;

printf("\nMultiplication of two complex numbers is: %d + %di",addimgrl,addimg);

}

void main () {

printf("Enter num 1 (real & imaginary): ");

scanf("%d %d",&num1.real , &num1.img);

printf("Enter num 2 (real & imaginary): ");

scanf("%d %d",&num2.real , &num2.img);

printf("\nNum 1 is: %d + %di\n",num1.real , num1.img);

printf("Num 2 is: %d + %di\n",num2.real , num2.img);

printf("\n\n\t\t\t\*MENU\*\n");

printf("\t1 to Add two complex numbers\n");

printf("\t2 to Subtract two complex numbers\n");

printf("\t3 to Multiply two complex numbers\n");

int ch;

printf("\nEnter your choice: ");

scanf("%d",&ch);

switch (ch) {

case 1: {

AddComplex(num1 , num2);

printf("\n");

break;

}

case 2: {

SubtractComplex(num1 , num2);

printf("\n");

break;

}

case 3: {

MultiplyComplex(num1 ,num2);

printf("\n");

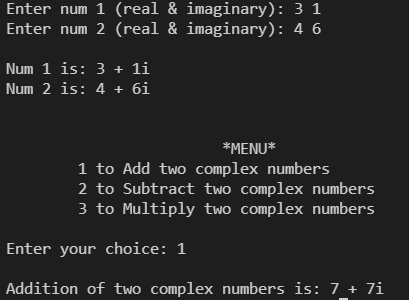
break;

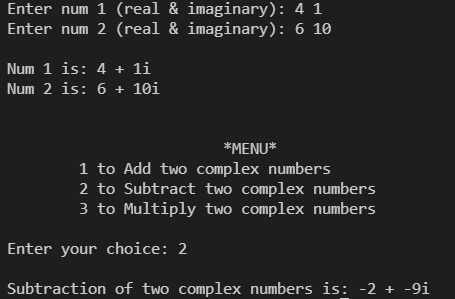
}

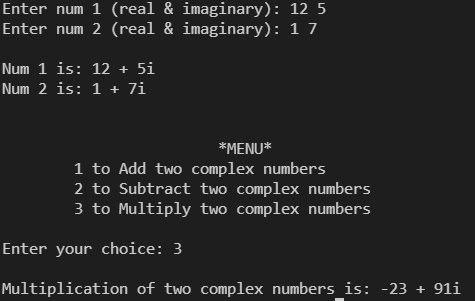
}

}

**Sample Output:**

****

****

****

Experiment-02

**Title: 1- Memory allocation**

**Objective:** To experiment the concepts of memory allocation

Problem - 2.1L: Create an integer array of user defined size n1 with dynamic memory allocation. Store data after reading from keyboard. Expand the size of array with n2. Read new values. Print state of array with all (n1+n2) values.

**Source Code:**

// A C program uses dynamic memory allocation to create and array and relocate memory when required.

#include <stdio.h>

#include <stdlib.h>

void main () {

int n1;

printf("Enter size of array: ");

scanf("%d",&n1);

int \*array;

// Dynamic memory allocated using malloc().

array = (int \*) malloc(n1\*sizeof(int));

for (int i=0; i<n1; i++) {

printf("Enter %d element: ",i+1);

scanf("%d",&array[i]);

}

int n2;

printf("\nEnter new size of array: ");

scanf("%d",&n2);

printf("\n");

// Memory reallocated dynamically using realloc().

array = realloc(array , n2\*sizeof(int));

for (int i=n1; i<n2; i++) {

printf("Enter %d element: ",i+1);

scanf("%d",&array[i]);

}

printf("\n");

for (int j=0; j<n2; j++) {

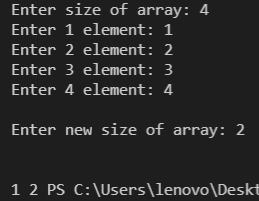
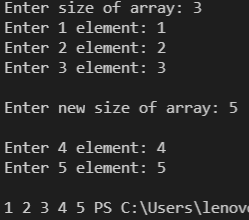
printf("%d ",array[j]);

}

free(array);

}

**Sample Output:**

**Title: 1- Memory allocation**

**Objective:** To experiment the concepts of memory allocation

Problem - 2.1L: Design a structure subject to store the details of the subjects like subject name and subject code. Using structure pointer allocate memory for the structure dynamically so as to obtain details of n subjects using for loop.

**Source Code:**

#include<stdio.h>

#include<stdlib.h>

struct subject{

int code;

char name[20];

};

void input(struct subject \*sub, int n){

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

printf("Enter subject code: ");

scanf("%d",&sub->code);

printf("Enter subject name: ");

scanf("%s",sub->name);

}

}

void display(struct subject \*sub, int n){

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

printf("\nSubject code: %d\n",sub->code);

printf("Subject name: %s\n",sub->name);

}

printf("\n");

}

void main(){

int n;

printf("Enter number of subjects: ");

scanf("%d",&n);

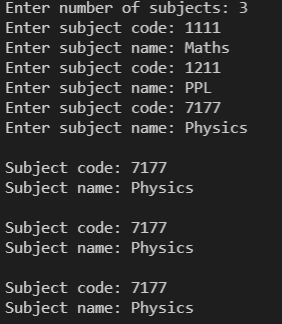
struct subject \*sub=(struct subject \*)malloc(n\*sizeof(struct subject));

input(sub,n);

display(sub,n);

}

**Sample Output:**

****

Experiment-03

**Title: Linked List Data Structure and its Applications**

**Objective:** To experiment the concept of pointers, structures and dynamic memory allocation to realize linked list and its application.

**Problem 3.1:** Implement single Linked List data structure and its operations like insert and delete in the beginning/end and nth position of the list, and display items stored items stored in linked list.

**Source Code:**

// A menu-driven C program which let's the user Insert , Delete , Display elements in list at different positions and situations.

#include <stdio.h>

#include <stdlib.h>

struct node {

int data;

struct node \*next;

};

struct node \*head;

// Structure used to create node again and again when required..

struct node \*CreateNode() {

struct node \*new = (struct node\*) malloc(sizeof(struct node));

return new;

}

void InsertAtBegin(int value) {

struct node \*NewNode = CreateNode();

if (head == NULL) { /\*Only works when list is empty\*/

NewNode->data = value;

head = NewNode;

NewNode->next = NULL;

} else {

printf("\n\t\*\*Element already exists at this position\*\*\n");

}

}

void InsertAtnthNode(int pos , int value) {

struct node\* temp = head;

if(pos==1) {

printf("\n\t\*\*Use Insert at begining\*\*\n");

} else {

struct node \*NewNode = CreateNode();

NewNode->data = value;

NewNode->next = NULL;

for (int i=0; i<pos-2; i++) {

temp = temp->next; /\*Accessing (n-1)th node\*/

}

NewNode->next = temp->next; /\*Linking nth node to (n+1)th node\*/

temp->next = NewNode; /\*Linking (n-1)th node to nth node\*/

}

}

void InsertAtEnd(int value) {

if (head == NULL) { /\*Does not work when list is empty. Underflow situation...\*/

printf("\n\t\*\*Use Insert at begining\*\*\n");

} else {

struct node \*temp = head;

while(temp->next!=NULL) {

temp = temp->next;

}

struct node \*NewNode = CreateNode();

NewNode->data = value;

NewNode->next = temp->next;

temp->next = NewNode; /\*Links new node n to (n-1)th node\*/

}

}

void DeleteAtBegin() {

if (head == NULL) { /\*Does not work when list is empty. Underflow situation...\*/

printf("\n\t\*\*No element exists\*\*\n");

} else {

head = head->next; /\*2nd node is now declared as head\*/

printf("\n\t\*\*Element deleted successfully\*\*\n");

}

}

void DeleteAtEnd() {

if (head == NULL) { /\*Does not work when list is empty. Underflow situation...\*/

printf("\n\t\*\*No element exists\*\*\n");

} else if (head->next == NULL) {

printf("\n\t\*\*Use Delete at begining\*\*\n");

} else {

struct node \*temp = head;

while(temp->next->next!=NULL) { /\*Accessing (n-1)th node\*/

temp = temp->next;

}

temp->next = NULL; /\*(n-1)th node will now point to null instead of nth node\*/

free(temp->next);

printf("\n\t\*\*Element deleted successfully\*\*\n");

}

}

void DeletenthNode(int pos) {

struct node \*temp = head;

if (pos == 1) {

printf("\n\t\*\*Use Delete at begining\*\*\n");

} else {

for (int i=0; i<pos-2; i++) {

temp = temp->next;

}

struct node \*temp2 = temp->next; /\*Accessing nth node, which we want to delete\*/

temp->next = temp2->next; /\*(n-1) node is pointing to (n+1) node now. Breaking the link between (n-1),n,(n+1) nodes.\*/

free(temp2);

printf("\n\t\*\*Element deleted successfully\*\*\n");

}

}

void Display() {

if (head == NULL) {

printf("\n\t\*\*No elements to display\*\*\n\n");

} else {

struct node \*temp = head;

printf("\nCurrent List:\n");

while(temp!=NULL) {

printf("%d ",temp->data);

temp = temp->next;

}

}

}

void main() {

head = NULL;

int ch;

while (1) {

printf("\n\t\*\*MENU\*\*\n\t1. Insert at begining\n\t2. Insert at nth position\n\t3. Insert at end\n\t4. Delete at begining\n\t5. Delete at end\n\t6. Delete nth node\n\t7. Display\n\t8. Exit\n");

printf("\n\tEnter your choice: ");

scanf("%d",&ch);

switch (ch) {

case 1:

printf("\nEnter value to be inserted: ");

int v1;

scanf("%d",&v1);

InsertAtBegin(v1);

break;

case 2:

printf("\nEnter position to insert value: ");

int v2 , pos1;

scanf("%d",&pos1);

printf("Enter value to be inserted: ");

scanf("%d",&v2);

InsertAtnthNode(pos1 , v2);

break;

case 3:

printf("\nEnter value to insert at end: ");

int v3;

scanf("%d",&v3);

InsertAtEnd(v3);

break;

case 4:

DeleteAtBegin();

break;

case 5:

DeleteAtEnd();

break;

case 6:

printf("\nEnter position to delete element: ");

int pos2;

scanf("%d",&pos2);

DeletenthNode(pos2);

break;

case 7:

Display();

break;

case 8:

printf("\n\t\*\*THANK YOU!\*\*\n");

exit(0);

default:

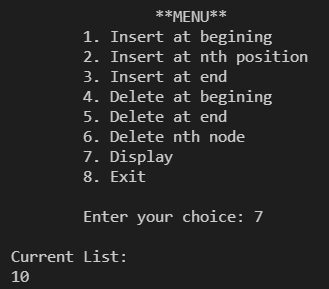
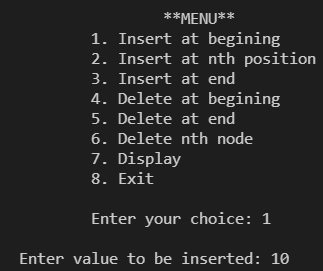
printf("\n\t\*\*Chose a valid option\*\*\n");

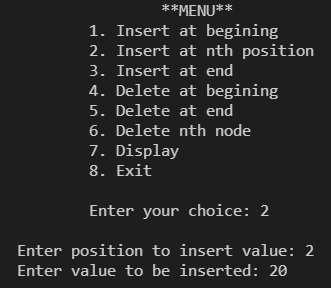
}

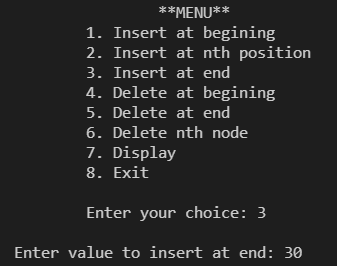
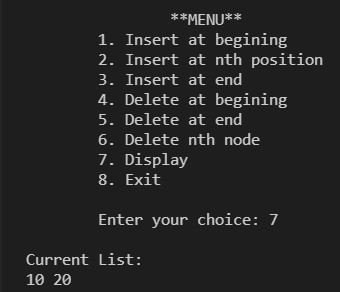
}

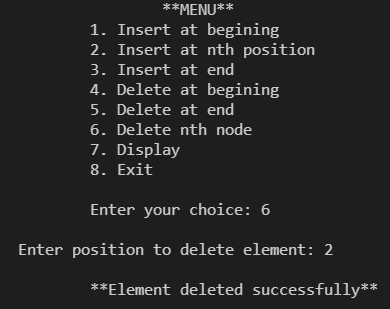
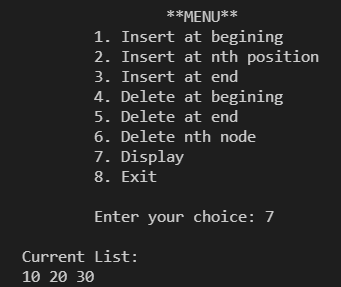
}

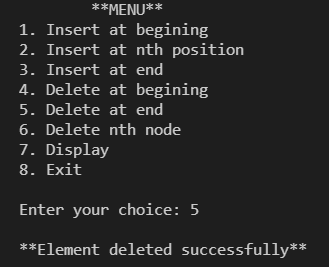
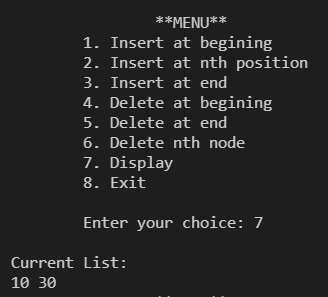
**Sample Output:­­­­­­­‑**

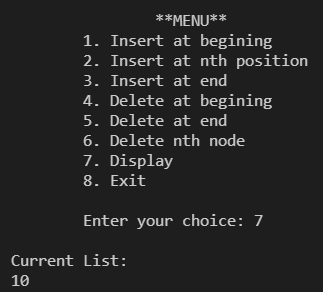
****

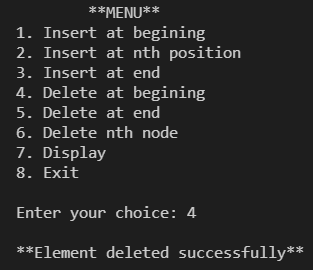
****

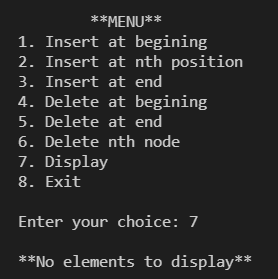
****

****

****

****

****

****

**Title: Linked List Data Structure and its Applications**

**Objective:** To experiment the concept of pointers, structures and dynamic memory allocation to realize linked list and its application.

**Problem 3.2:** Using singly linked list and functions implement Stack and its operations like insert, delete and display.

**Source Code:**

// 31st January, 2022.

// Lab Activity - 3, Question - 2.

// Mridul Vasudeva

// A C program to implement stack using linked list.

#include <stdio.h>

#include <stdlib.h>

struct node {

int data;

struct node \*next;

};

struct node\* top = NULL;

struct node\* CreateNode() {

struct node\* new = (struct node\*) malloc(sizeof(struct node));

return new;

}

void push(int value) {

struct node\* NewNode = CreateNode();

NewNode->data = value;

NewNode->next = top;

top = NewNode;

}

void pop() {

struct node\* temp = top;

if(temp == NULL) {

printf("\n\t\*\*Stack Underflow\*\*\n");

return;

}

top = temp->next;

printf("\n\t\*\*%d deleted successfully\*\*\n",temp->data);

free(temp);

}

void Top() {

if(top==NULL) {

printf("\n\t\*\*Stack Underflow\*\*\n");

return;

}

printf("\n\t\*\*Top Element: %d\*\*\n",top->data);

}

void print() {

struct node\* temp = top;

while(temp!=NULL) {

printf("%d ",temp->data);

temp = temp->next;

}

}

void main() {

int choice, item;

while(1) {

printf("\n\t\t\*\*MENU\*\*\n\t1. Insert an element\n\t\2. Delete an element\n\t3. Find top element\n\t4. View Stack\n\t5. Exit\n");

printf("Enter your choice: ");

scanf("%d",&choice);

switch(choice) {

case 1:

printf("\nEnter value to be inserted: ");

scanf("%d",&item);

push(item);

break;

case 2:

pop();

break;

case 3:

Top();

break;

case 4:

print();

break;

case 5:

exit(0);

default:

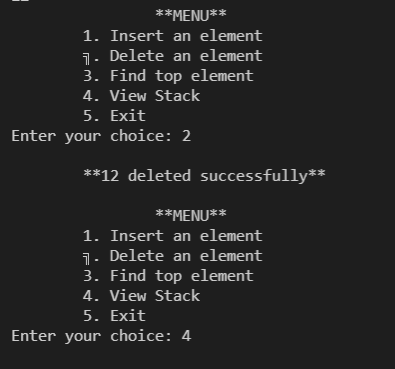
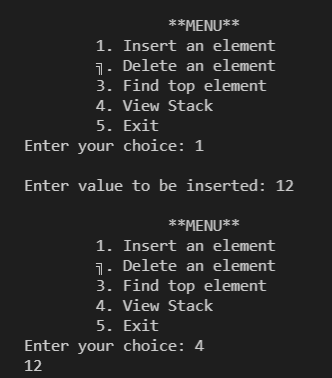
printf("\n\t\*\*Enter a valid choice\*\*\n");

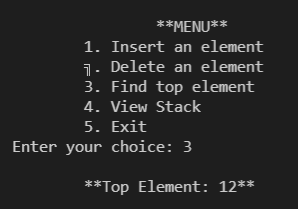
}

}

}

**Sample Output:**

****

****

**Practice Activity – 3**

**Title: Linked List Data Structure and its Applications**

**Objective:** To experiment the concept of pointers, structures and dynamic memory allocation to realize linked list and its application.

**Problem 3.1P:** Add two polynomials using linked list.

**Source Code:**

#include<stdio.h>

#include<stdlib.h>

struct node{

int coeff;

int exp;

struct node \*link;

};

struct node \* create\_new(){

return NULL;

}

struct node \* create\_node(){

return ((struct node \*)malloc(sizeof(struct node)));

}

void display(struct node \*start){ //traversing

struct node \*temp=start;

if (temp==NULL){

printf("Linked list is empty.\n");

return;

}

while (temp!=NULL){

printf("%dx^%d",temp->coeff, temp->exp);

if(temp->link!=NULL) {

printf(" + ");

}

temp=temp->link;

}

}

void insert(struct node \*\*start, int coeff, int exp){ //find the last node and update the link of last node with new node

struct node \*temp=\*start;

struct node \*new=create\_node();

if (\*start==NULL){

new->link=\*start;

new->coeff=coeff;

new->exp=exp;

\*start=new;

return;

}

while (temp->link!=NULL)

temp=temp->link;

new->coeff=coeff;

new->exp=exp;

new->link=NULL;

temp->link=new;

}

void create\_poly(struct node \*\*start){

int con=0;

do{

int coeff,exp;

printf("\nEnter degree: ");

scanf("%d",&exp);

printf("Enter coefficient: ");

scanf("%d",&coeff);

insert(start,coeff,exp);

printf("\nPress 1 to continue/0 to Exit: ");

scanf("%d",&con);

}while(con==1);

}

void add(struct node \*p1, struct node \*p2, struct node \*\*res){

int coeff,exp;

while(p1 && p2){

if (p1->exp>p2->exp){

insert(res, p1->coeff, p1->exp);

p1=p1->link;

}else if (p1->exp<p2->exp){

insert(res,p2->coeff,p2->exp);

p2=p2->link;

}else{

coeff=p1->coeff+p2->coeff;

insert(res,coeff,p2->exp);

p1=p1->link;

p2=p2->link;

}

}

while (p1){

insert(res,p1->coeff,p1->exp);

p1=p1->link;

}

while (p2){

insert(res,p2->coeff,p2->exp);

p2=p2->link;

}

}

void main(){

struct node \*poly1=create\_new();

struct node \*poly2=create\_new();

struct node \*result=create\_new();

create\_poly(&poly1);

printf("Polynomial 1: ");

display(poly1);

create\_poly(&poly2);

printf("Polynomial 2: ");

display(poly2);

add(poly1,poly2,&result);

printf("\n\n");

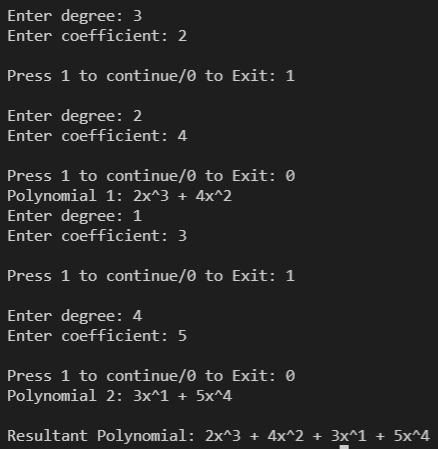
printf("Resultant Polynomial: ");

display(result);

printf("\n");

}

**Sample Output:**



**Title: Linked List Data Structure and its Applications**

**Objective:** To experiment the concept of pointers, structures and dynamic memory allocation to realize linked list and its application.

**Problem 3.2P:** Implement Circular linked list.

**Source Code:**

// Implementing Circular linked list

#include <stdio.h>

#include <stdlib.h>

struct node{

int data;

struct node \*next;

};

struct node \*last;

struct node \*CreateNode() {

struct node \*new = (struct node\*) malloc(sizeof(struct node));

return new;

}

void Insert(int item) {

struct node \*NewNode = CreateNode(); /\*Allocating memory to new node\*/

if(last==NULL) { /\*When list is empty\*/

NewNode->data = item; /\*Assign data value to this new node\*/

NewNode->next = NewNode; /\*Since we have only one node, it would point agian to itself\*/

last = NewNode; /\*Last now points to NewNode\*/

} else{

NewNode->data = item;

NewNode->next = last->next;

last->next = NewNode;

}

}

void Display() {

struct node \*temp = last->next;

printf("\nList: ");

do {

printf("%d ",temp->data);

temp = temp->next;

} while(temp!=last->next);

printf("\n");

}

void main() {

int n, ele;

printf("Enter number of elements: ");

scanf("%d",&n);

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

printf("Enter element: ");

scanf("%d",&ele);

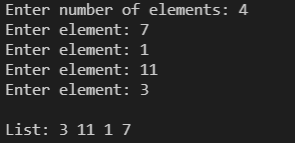
Insert(ele);

}

Display();

}

**Sample Ouput:**

****

**Title: Linked List Data Structure and its Applications**

**Objective:** To experiment the concept of pointers, structures and dynamic memory allocation to realize linked list and its application.

**Problem 3.3P:** Implement Doubly linked list.

**Source Code:**

// Implementing Doubly linked list.

#include <stdio.h>

#include <stdlib.h>

struct Node {

int value;

struct Node \*next;

struct Node \*prev;

};

struct Node \*head;

struct Node \*CreateNode() {

struct Node \*new = (struct Node\*) malloc(sizeof(struct Node));

return new;

}

void Insert(int val) { /\*Inserting element at head\*/

struct Node \*NewNode = CreateNode(); /\*NewNode is created everytime function is called\*/

NewNode->value = val; /\*Value assigned to NewNode\*/

NewNode->next = head; /\*NewNode's next points to head\*/

NewNode->prev = NULL; /\*NewNode's previous points to NULL\*/

if (head != NULL) {

head->prev = NewNode;

}

head = NewNode;

}

void Display() {

struct Node \*temp = head;

printf("\nForward:\n"); /\*Printing normally in forward manner\*/

while(temp!=NULL) {

printf("%d ",temp->value);

temp = temp->next;

}

}

void ReverseDisplay() {

struct Node \*temp = head;

while(temp->next!=NULL) { /\*Moving to the last node\*/

temp = temp->next;

}

printf("\nBackward:\n"); /\*Printing in backward manner\*/

while(temp!=NULL) {

printf("%d ",temp->value);

temp = temp->prev;

}

printf("\n");

}

void main() {

int n, val;

printf("Enter number of elements: ");

scanf("%d",&n);

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

printf("Enter element: ");

scanf("%d",&val);

Insert(val); /\*Inserting value everytime loop executes\*/

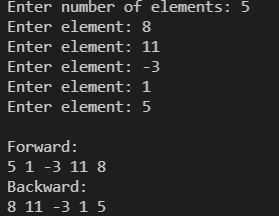
}

Display();

ReverseDisplay();

}

**Sample Output:**

****

Experiment-04

**Title: Stack Data Structure**

**Objective:** To demonstrate the use of arrays, and linked list to implement stack operations and applications of stack.

**Problem 4.1:** Using array and functions implement Stack and its operations like push, pop, peek.

**Source Code:**

// A C program which implements the concept of Stack(Last in First out) data structure.

#include <stdio.h>

#include <stdlib.h>

// top element and array are globally defined so they can be used anywhere in the program.

static int top = -1;

int array[25];

void push(int item) {

top++;

\*(array+top) = item;

}

void pop() {

if(top == -1) {

printf("\n\t\*\*Stack Underflow\*\*\n");

return;

}

top--;

printf("\n\t\*\*%d deleted successfully\*\*\n",array[top+1]);

}

void Top() {

if (top == -1) {

printf("\n\t\*\*Stack Underflow\*\*\n");

return;

}

printf("\n\t\*\*Top Element: %d\*\*\n",array[top]);

}

void print() {

printf("\nStack: ");

if (top == -1) {

printf("Empty");

return;

}

for (int i=0; i<=top; i++) {

printf("%d ",array[i]);

}

}

void main() {

int choice, item;

while(1) {

printf("\n\t\t\*\*MENU\*\*\n\t1. Insert an element\n\t\2. Delete an element\n\t3. Find top element\n\t4. View Stack\n\t5. Exit\n");

printf("Enter your choice: ");

scanf("%d",&choice);

switch(choice) {

case 1:

printf("\nEnter value to be inserted: ");

scanf("%d",&item);

push(item);

break;

case 2:

pop();

break;

case 3:

Top();

break;

case 4:

print();

break;

case 5:

exit(0);

default:

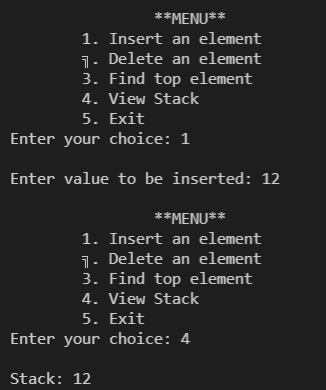
printf("\n\t\*\*Enter a valid choice\*\*\n");

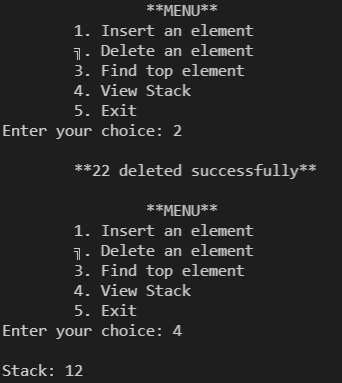
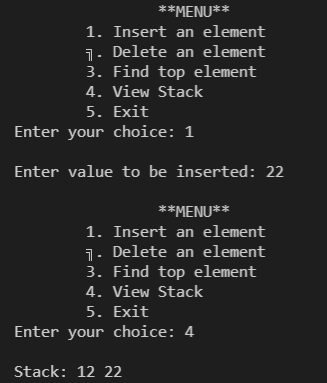
}

}

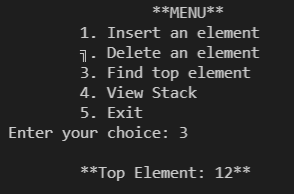
}

**Sample Output:**

****

****

**­**

****

**Title: Stack Data Structure**

**Objective:** To demonstrate the use of arrays, and linked list to implement stack operations and applications of stack.

**Problem 4.2:** Use the concept of Stack to reverse a string.

**Source Code:**

// A C program to reverse a string using stack.

#include <stdio.h>

#include <stdlib.h>

static int top = -1;

char array[50];

void push(int item) {

top++;

\*(array+top) = item;

}

int pop() {

return array[top--];

}

void main() {

char ch[50];

printf("Enter a string: ");

gets(ch);

int i = 0;

while(ch[i] != '\0') {

push(ch[i]);

i++;

}

for(int i=0; ch[i] != '\0'; i++) {

ch[i] = pop();

}

printf("Reversed String: ");

puts(ch);

}

**Sample Output:**

**C2.1.PNGC2.2.PNG**

**Title: Stack Data Structure**

**Objective:** To demonstrate the use of arrays, and linked list to implement stack operations and applications of stack.

**Problem 4.1P:** Using array and functions to implement two Stacks and its operations(push, pop and peek).

**Source Code:**

#include <stdio.h>

#include <stdlib.h>

#define Max 100

int array[Max];

// Stack 1: array[Max/2] = array[100/2] = array[50] Memory for stack 1 till index 50...

// Stack 2: array[(Max/2)+1] = array[(100/2)+1] = array[51] Memory for stack 2 from 51-100...

static int top1 = -1;

static int top2 = 50;

void push1(int item1) {

if(top1 == Max/2) {

return;

} else {

top1++;

array[top1] = item1;

}

}

int pop1() {

return array[top1];

top1--;

}

void push2(int item2) {

if(top2 == Max) {

return;

} else {

top2++;

array[top2] = item2;

}

}

int pop2() {

return array[top2];

top2--;

}

void main() {

int ch, i1, i2, p1, p2;

while(1) {

printf("\n\t\t\*\*MENU\*\*\n\t1. Insert in Stack 1\n\t2. Insert in Stack 2\n\t3. Delete from Stack 1\n\t4. Delete from Stack 2\n\t5. Exit\n");

printf("\n\tEnter your choice: ");

scanf("%d",&ch);

switch(ch) {

case 1:

printf("\nEnter element to be inserted: ");

scanf("%d",&i1);

push1(i1);

break;

case 2:

printf("\nEnter element to be inserted: ");

scanf("%d",&i2);

push2(i2);

break;

case 3:

p1 = pop1();

printf("\n\*\*%d Deleted from Stack 1\*\*\n",p1);

break;

case 4:

p2 = pop2();

printf("\n\*\*%d Deleted from Stack 2\*\*\n",p2);

break;

case 5:

exit(0);

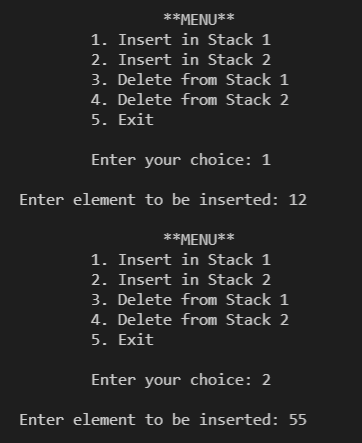
break;

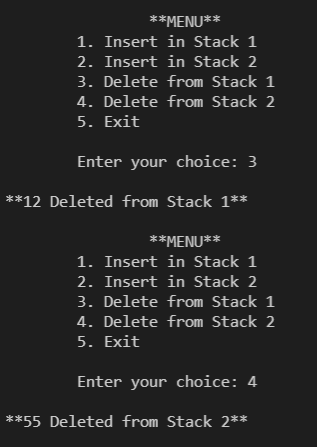
}

}

}

**Sample Output:**

****

****

**Title: Stack Data Structure**

**Objective:** To demonstrate the use of arrays, and linked list to implement stack operations and applications of stack.

**Problem 4.2P:** Convert infix to prefix expression using stack and array.

**Source Code:**

#include<stdio.h>

#include<string.h>

int TOP=-1;

void Push(char \*stack, char ins, int max)

{

if (TOP==max)

printf("Overflow.");

else{

TOP=TOP+1;

stack[TOP]=ins;

}

}

char Pop(char \*stack)

{

if (TOP==-1)

printf("Underflow.");

else{

char item=stack[TOP];

TOP=TOP-1;

return item;

}

}

char View(char \*stack){

if (TOP==-1) printf("Empty stack.");

else

return stack[TOP];

}

void POST(char \*infix, char \*post){

int count=0;

int max=strlen(infix);

char stack[max],temp;

Push(stack,'(',max);

for (int i=0; TOP!=-1; i++){

if (infix[i]>64 && infix[i]<91){

post[count]=infix[i];

count=count+1;

} else if (infix[i]=='(')

Push(stack, infix[i], max);

else if (infix[i]=='+' || infix[i]=='-' || infix[i]=='\*' || infix[i]=='/' || infix[i]=='^'){

if (View(stack)=='(')

Push(stack, infix[i], max);

else{

if (infix[i]=='+' || infix[i]=='-'){

while (View(stack)=='^' || View(stack)=='\*' || View(stack)=='/' || View(stack)=='+' || View(stack)=='-'){

post[count]=Pop(stack);

count++;

}

}else if (infix[i]=='\*' || infix[i]=='/'){

while (View(stack)=='^' || View(stack)=='\*' || View(stack)=='/'){

post[count]=Pop(stack);

count++;

}

} else{

while (View(stack)=='^') {

post[count]=Pop(stack);

count++;

}

}

Push(stack,infix[i],max);

}

} else if (infix[i]==')'){

while (View(stack)!='('){

post[count]=Pop(stack);

count++;

}

temp=Pop(stack);

}

}

}

void main(){

char infix[]="Z\*(X+Y-(M/N\*O)^P)/Q)";

int n=strlen(infix);

char post[n];

POST(infix,post);

printf("%s",post);

printf("\n");

}

**Sample Output:**

**C2..PNG**

**Title: Stack Data Structure**

**Objective:** To demonstrate the use of arrays, and linked list to implement stack operations and applications of stack.

**Problem 4.3P:** Evaluate postfix expression using stack and array.

**Sample Output:**

#include<stdio.h>

#include<string.h>

int TOP = -1;

void PUSH(char \*stack, char ins, int max){

if (TOP==max)

printf("Overflow.");

else{

TOP=TOP+1;

stack[TOP]=ins;

}

}

char POP(char \*stack){

if (TOP==-1)

printf("Underflow.");

else {

char item=stack[TOP];

TOP=TOP-1;

return item;

}

}

int eval(char \*post){

int max=strlen(post);

char stack[max],ans,A,B;

int res;

for (int i=0; post[i]!=')'; i++){

if (post[i]>47 && post[i]<58)

PUSH(stack,post[i],max);

else{

if (post[i]=='+'){

A=POP(stack);

B=POP(stack);

ans=(int)B+(int)A;

PUSH(stack,ans,max);

} else if (post[i]=='-'){

A=POP(stack);

B=POP(stack);

ans=(int)B-(int)A;

PUSH(stack,ans,max);

} else if (post[i]=='\*'){

A=POP(stack);

B=POP(stack);

ans=(int)B\*(int)A;

PUSH(stack,ans,max);

} else if (post[i]=='/'){

A=POP(stack);

B=POP(stack);

ans=(int)B/(int)A;

PUSH(stack,ans,max);

}

}

}

res=(int)POP(stack);

return res;

}

void main(){

char post[]={'8','1','2','+','\*','7','1','/','-',')'};

int ans=eval(post);

printf("%d \n",ans);

}

**Sample Output:**

**C3.PNG**

Experiment-05

**Title: Queue Data Structure**

**Objective:** To demonstrate use of arrays and linked list to implement Queue operations and types of queues.

**Problem 5.1:** Using array and functions implement Queue data structure and its operations like insert, delete.

**Source Code:**

// Implementing Queue using Array.

#include <stdio.h>

#include <stdlib.h>

static int front = 0;

static int rear = -1;

void Dequeue(int \*array) {

printf("\n\*\*%d deleted\*\*\n",\*(array+front));

front++;

}

void Enqueue(int item, int \*array) {

rear++;

\*(array+rear) = item;

printf("\n\*\*%d inserted\*\*\n",\*(array+rear));

}

void ViewQueue(int \*array) {

printf("Queue: ");

for (int i=front; i<=rear; i++) {

printf("%d ",\*(array+i));

}

printf("\n");

}

void main() {

int array[10], ch, data;

while (1) {

printf("\n\t\t\*\*MENU\*\*\n\t1. Insert element in Queue\n\t2. Delete element from Queue\n\t3. View Queue\n\t4. Exit\n");

printf("\nEnter your choice: ");

scanf("%d",&ch);

switch(ch) {

case 1:

printf("Enter element to be inserted: ");

scanf("%d",&data);

Enqueue(data,array);

break;

case 2:

Dequeue(array);

break;

case 3:

ViewQueue(array);

break;

case 4:

exit(0);

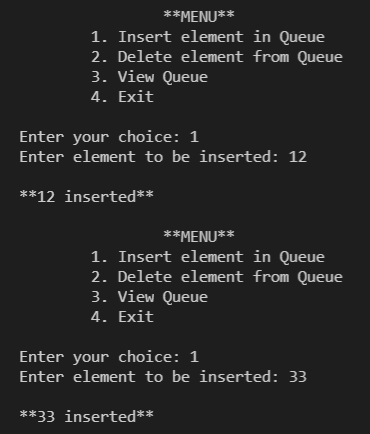
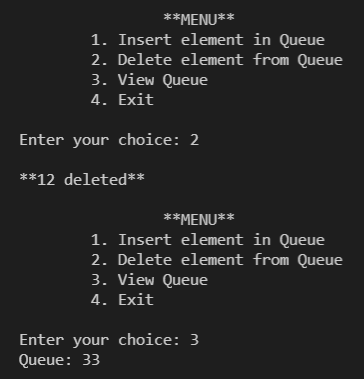
break;

}

}

}

**Sample Output:**

****

**Title: Queue Data Structure**

**Objective:** To demonstrate use of arrays and linked list to implement Queue operations and types of queues.

**Problem 5.2:** Check whether the string is palindrome or not Stack and Queue.

**Sample Output:**

// Check if a string is palindrome or not...

#include <stdio.h>

#define max 50

static int top = -1;

static int front = 0;

static int rear = -1;

char StackArray[max];

char QueueArray[max];

void push(char e) {

StackArray[top++] = e;

}

char pop() {

return StackArray[top--];

}

void Enqueue(char e) {

QueueArray[rear++] = e;

}

char Dequeue() {

return QueueArray[front];

front++;

}

void IsPallindrome(char \*str) {

for(int i=0; i<='\0'; i++) {

push(\*(str+i));

Enqueue(\*(str+i));

}

for(int i=0; i<='\0'; i++) {

char Sp = pop();

char Qd = Dequeue();

if(Sp == Qd) {

printf("It is pallindrome\n");

} else {

printf("It is not pallindrome\n");

}

}

}

void main() {

char str[max];

printf("Enter a string: ");

gets(str);

IsPallindrome(str);

}

**Source Output:**

**C2.PNG**

**Practice Activity - 5**

**Title: Queue Data Structure**

**Objective:** To demonstrate use of arrays and linked list to implement Queue operations and types of queues.

**Problem 5.1P:** Implement Queue data structure using linked list and its operations(Enqueue, Dequeue, Display).

**Source Code:**

// Implementing Queue using linked list...

#include <stdio.h>

#include <stdlib.h>

typedef struct node{

int data;

struct node \*next;

}node;

node \*front = NULL;

node \*rear = NULL;

struct node \*CreateNode() {

node \*new = (node\*) malloc(sizeof(node));

return new;

}

void Enqueue(int item) {

node \*NewNode = CreateNode();

NewNode->data = item;

NewNode->next = NULL;

if(front == NULL && rear == NULL) {

front = rear = NewNode;

return;

}

rear->next = NewNode;

rear = NewNode;

}

void Dequeue() {

node \*temp = front;

if(front == NULL) {

return;

} else {

printf("\n\*\*%d Dequeued\*\*\n",front->data);

front = front->next;

}

free(temp);

}

void Display() {

node \*temp = front;

while(temp!=NULL) {

printf("%d ",temp->data);

temp = temp->next;

}

}

void main() {

int ch, val;

while(1) {

printf("\n\t\t\*\*MENU\*\*\n\t1. Enqueue\n\t2. Dequeue\n\t3. Display\n\t4. Exit\n");

printf("Enter your choice: ");

scanf("%d",&ch);

switch(ch) {

case 1:

printf("Enter value to be inserted: ");

scanf("%d",&val);

Enqueue(val);

break;

case 2:

Dequeue();

break;

case 3:

Display();

break;

case 4:

exit(0);

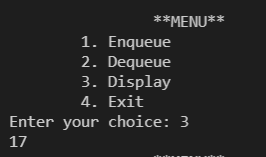
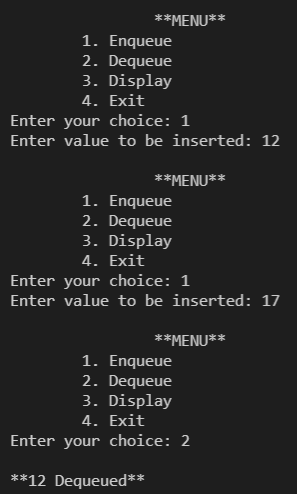
break;

}

}

}

**Sample Output:**

****