1. **Activity:** Find sum of all array elements using recursion

**Algorithm:**

1. Get number of elements in the array
2. Create a new array of the given size and get the elements from the user
3. Call the function to fin sum of elements reclusively passing array and the length (last index + 1):
   1. Check if the passed array has length <= 0, if yes return 0 as the array is empty
   2. If not recursively call the function in the return statement also adding the element at the index-1 passed by the recursive call before it.
4. Get the final sum at the final call after which the function get exits completely, store this value in a variable.
5. Print the sum.

**Test Case Table:**

|  |  |  |  |
| --- | --- | --- | --- |
| Array Length | Elements entered in array | Sum printed out | Pass/Fail |
| 5 | {1,2,3,4,5} | 15 | Pass |
| 2 | {1,5} | 6 | Pass |
| 10 | {1,6,3,7,2,10,7,3,6,8} | 53 | Pass |
| 6 | {1,1,1,1,1,1} | 6 | Pass |

**Program:**

*#include* <stdio.h>

*#include* <stdlib.h>

*//Function returns the received value + current array element*

int findsum(int arr[], int n){

*//If array is empty return 0*

*if*(n <= 0){

*return* 0;

    }

*//Recursive call*

*return* (findsum(arr, n-1) + arr[n-1]);

}

int main(){

    printf("Enter number of elements: ");

*//Get size of array and create an array with the entered size*

    int n;

    scanf("%d", &n);

    int \*arr = (int\*)malloc(n\*sizeof(int));

    printf("Enter elements:\n");

*//Get elements*

*for* (int i = 0; i < n; i++)

    {

        scanf("%d", arr + i);

    }

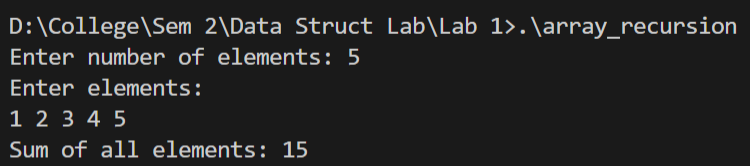
    int sum = findsum(arr, n);

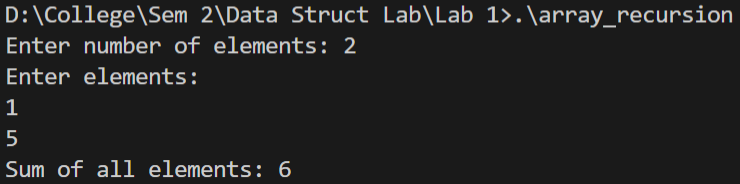
    printf("Sum of all elements: %d", sum);

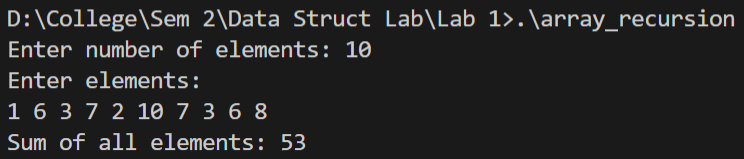
*return* 0;

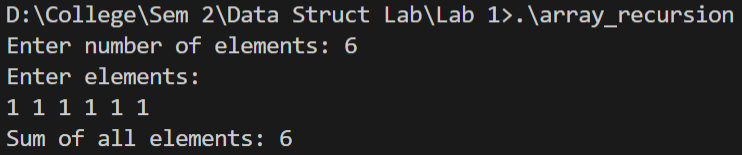
}

**Screenshot of compilation and execution:**

****

****

****

****

1. **Activity:** Create a menu driven program that works with an integer array ‘a1’ …….. display() gets a reference of any array and prints the content and count.

**Program:**

*#include*<stdio.h>

*#include* <ctype.h>

*#include*<stdlib.h>

void clear (void) *//Clears the buffer memory*

{

*while* ( getchar() != '\n' );

}

int input;

int arrayA[10], arrayB[10], arrayC[20];

int countA=0, countB=0, countC=0;

void MainMenu();

void Delete(int\*, int\*, char);

void Copy(int\*, int, int\*, int\*);

void Merge(int\*, int, int\*, int, int\*, int\*);

void Display(int\*, int, char);

void Insert(int\*, int\*, char);

void Merge(int\* sArray1, int sArray1Size, int\* sArray2, int sArray2Size, int\* dArray, int\* dArraySize)

{

    system("cls");

*if*(sArray1Size+sArray2Size<=0) *//Checking if both array are empty or not*

    {

        printf("\nBoth arrays are empty.\nPress any key to continue...");

        clear(); getchar();

*return*;

    }

    int i;

*for*(i=0; i<sArray1Size; i++) *//Copying the contents of 1st array to 3rd array*

    {

        dArray[i]=sArray1[i];

    }

*for*(i=sArray1Size; i<sArray1Size+sArray2Size; i++) *//Copying the contents of 2nd array to 3rd array*

    {

        dArray[i]=sArray2[i-sArray1Size];

    }

    (\*dArraySize)=sArray1Size+sArray2Size;

*return*;

}

void Copy(int\* sArray, int sArraySize, int\* dArray, int\* dArraySize)

{

    system("cls");

*if*((\*dArraySize)+sArraySize>10) *//Checking for Overflow*

    {

        printf("\nCannot copy! Err: Data Overflow! Press any key to continue...");

        clear(); getchar();

*return*;

    }

*if*(sArraySize<=0) *//Checking if the source array is empty*

    {

        printf("\nSource array is empty! Press any key to continue...");

        clear(); getchar();

*return*;

    }

*if*((\*dArraySize)<=0) *//If destination array is empty, copy directly*

    {

*for*(int i=0; i<sArraySize; i++)

        {

            dArray[i]=sArray[i];

        }

*return*;

    }

*else* *//Otherwise ask user for following*

    {

        Copy\_Menu\_F:

        printf("\nThere are some elements in the destination array.");

        printf("\n1) Overwrite.");

        printf("\n2) Insert to left");

        printf("\n3) Insert to right");

        printf("Input: ");

        clear();

        scanf("%d", &input);

*switch* (input)

        {

*case* 1:

*for*(int i=0; i<sArraySize; i++) *//Overwriting*

                {

                    dArray[i]=sArray[i];

                }

*if*(sArraySize>(\*dArraySize)) *//Checking if there is some old data left after overwriting*

                {

                    (\*dArraySize)=sArraySize;

                }

*return*;

*break*;

*case* 2:

*for*(int i=0; i<(\*dArraySize); i++) *//Shifting old data towards right*

                {

                    dArray[sArraySize+i]=dArray[i];

                }

*for*(int i=0; i<sArraySize; i++) *//Copying from source array*

                {

                    dArray[i]=sArray[i];

                }

                (\*dArraySize)=(\*dArraySize)+sArraySize; *//Adjusting size parameter*

*return*;

*break*;

*case* 3:

*for*(int i=0; i<sArraySize; i++) *//Copying from source array*

                {

                    dArray[(\*dArraySize)+i]=sArray[i];

                }

                (\*dArraySize)=(\*dArraySize)+sArraySize; *//Adjusting size parameter*

*return*;

*break*;

*default*:

                printf("Wrong Input! Press any key to try again..."); *//Checking for invalid input*

                clear(); getchar();

                system("cls");

*goto* Copy\_Menu\_F;

        }

    }

}

void Delete(int\* array, int\* arraySize, char name)

{

    int pos;

    Delete\_Menu\_F:

    system("cls");

*if*(\*arraySize<=0) *//Checking for underflow*

    {

        printf("Data Underflow!\nPress any key to continue...");

        clear(); getchar();

*return*;

    }

    Display(array, \*arraySize, name);

    printf("\nEnter the position you want to delete(1-10): ");

    scanf("%d", &pos);

*if*(pos>10 || pos<1) *//Checking if position is valid or not (in range of 1-10 including both)*

    {

        printf("\nWrong Input!\nPress any key to try again...");

        clear(); getchar();

*goto* Delete\_Menu\_F;

    }

*if*(pos>\*arraySize) *//Checking if there exists any element at the position*

    {

        printf("\nNo element at position %d.\nPress any key to continue...", pos);

        clear(); getchar();

*return*;

    }

    pos -= 1;

*for*(int i=pos; i<\*arraySize; i++) *//Shifting data to overwrite the required element*

    {

        array[i]=array[i+1];

    }

    (\*arraySize) -= 1; *//Adjusting size parameter*

    system("cls");

}

void Insert(int\* array, int\* arraySize, char name)

{

    int position, value, i;

    printf("Current occupancy of array is: %d", countA);

*if*(\*arraySize>=10) *//Checking for overflow*

    {

        printf("\nData overflow! Cannot store more data in Array A.\nPress any key to continue...");

        clear(); getchar();

        system("cls");

        MainMenu();

    }

    Insert\_Menu\_F:

    printf("\nEnter the position where you want to insert the element(1-10): ");

    scanf("%d", &position);

*if*(position>10 || position<1) *//Checking if position is valid or not (in range of 1-10 including both)*

    {

        printf("\nInvalid position entered! Press any key to try again...");

        clear(); getchar();

        system("cls");

*goto* Insert\_Menu\_F;

    }

    position -= 1;

    printf("Enter the element to store: ");

    scanf("%d", &value);

*if*(position<=\*arraySize) *//If user wants to add in between existing data*

    {

*for*(i=(\*arraySize); i>position; i--) *//Shifting elements*

        {

            array[i]=array[i-1];

        }

        array[position]=value;

    }

*else* *//If user wants to add at the end or beyond existing data*

    {

        position=(\*arraySize);

        array[position]=value;

    }

    ++(\*arraySize);

    system("cls");

}

void Display(int\* array, int arraySize, char name)

{

    system("cls");

    int i;

    printf("\nNumber of elements in array %c: %d", name, arraySize);

    printf("\nArray:");

*for*(i=0; i<arraySize; i++)

    {

        printf("%d ", array[i]);

    }

    printf("\n\nPress any key to continue...");

    clear(); getchar();

}

void MainMenu()

{

    system("cls");

    printf("       Main Menu\n-----------------------");

    printf("\n1) Display array");

    printf("\n2) Insert an element");

    printf("\n3) Delete an element");

    printf("\n4) Copy an array");

    printf("\n5) Merge arrays");

    printf("\n6) Exit");

    printf("\n\n( Arrays available: 3 )");

    printf("\n\nInput: ");

    scanf("%d", &input);

*switch* (input)

    {

*case* 1:

            system("cls");

            Display\_Menu: *//Sub menu for display*

            printf("     Display menu\n-----------------------\n   Select an array.\n-----------------------");

            printf("\n1) Array 1");

            printf("\n2) Array 2");

            printf("\n3) Array 3");

            printf("\n4) Go back.");

            printf("\n\nInput: ");

            clear();

            scanf("%d", &input);

*switch* (input)

            {

*case* 1:

                    system("cls");

                    Display(arrayA, countA, 'A');

                    MainMenu();

*break*;

*case* 2:

                    system("cls");

                    Display(arrayB, countB, 'B');

                    MainMenu();

*break*;

*case* 3:

                    system("cls");

                    Display(arrayC, countC, 'C');

                    MainMenu();

*break*;

*case* 4:

                    system("cls");

                    MainMenu();

*break*;

*default*:

                    system("cls");

                    printf("Wrong Input!\n\n");

*goto* Display\_Menu;

            }

*break*;

*case* 2:

            Insert\_Menu: *//Sub menu for insert*

            system("cls");

            printf("    Insert Menu\n-----------------------\n  Select an array.\n-----------------------");

            printf("\n1) Array 1");

            printf("\n2) Array 2");

            printf("\n3) Array 3");

            printf("\n4) Go back.");

            printf("\n\nInput: ");

            clear();

            scanf("%d", &input);

*switch* (input)

            {

*case* 1:

                    system("cls");

                    Insert(arrayA, &countA, 'A');

                    Display(arrayA, countA, 'A');

                    MainMenu();

*break*;

*case* 2:

                    system("cls");

                    Insert(arrayB, &countB, 'B');

                    Display(arrayB, countB, 'B');

                    MainMenu();

*break*;

*case* 3:

                    system("cls");

                    Insert(arrayC, &countC, 'C');

                    Display(arrayC, countC, 'C');

                    MainMenu();

*break*;

*case* 4:

                    MainMenu();

*break*;

*default*:

                    printf(" Wrong Input! Press any key to try again...");

                    clear(); getchar();

*goto* Insert\_Menu;

            }

*break*;

*case* 3:

                Delete\_Menu:

                system("cls"); *//Sub menu for delete*

                printf("     Delete Menu\n-----------------------\n  Select an arrray.\n-----------------------");

                printf("\n1) Array 1");

                printf("\n2) Array 2");

                printf("\n3) Array 3");

                printf("\n4) Go back.");

                printf("\n\nInput: ");

                clear();

                scanf("%d", &input);

*switch* (input)

                {

*case* 1:

                        system("cls");

                        Delete(arrayA, &countA, 'A');

                        Display(arrayA, countA, 'A');

                        MainMenu();

*break*;

*case* 2:

                        system("cls");

                        Delete(arrayB, &countB, 'B');

                        Display(arrayB, countB, 'B');

                        MainMenu();

*break*;

*case* 3:

                        system("cls");

                        Delete(arrayC, &countC, 'C');

                        Display(arrayC, countC, 'C');

                        MainMenu();

*break*;

*case* 4:

                        MainMenu();

*break*;

*default*:

                        printf(" Wrong Input! Press any key to try again...");

                        clear(); getchar();

*goto* Delete\_Menu;

            }

*break*;

*case* 4:

                Copy\_Menu:

                system("cls"); *//Sub menu for Copy*

                printf("      Copy Menu\n-----------------------\n  Select an option.\n-----------------------");

                printf("\n1) Copy Array 1 to Array 2");

                printf("\n2) Copy Array 2 to Array 1");

                printf("\n3) Go back.");

                printf("\n\nInput: ");

                clear();

                scanf("%d", &input);

*switch*(input)

                {

*case* 1:

                        system("cls");

                        Copy(arrayA, countA, arrayB, &countB);

                        Display(arrayB, countB, 'B');

                        MainMenu();

*break*;

*case* 2:

                        system("cls");

                        Copy(arrayB, countB, arrayA, &countA);

                        Display(arrayA, countA, 'A');

                        MainMenu();

*break*;

*case* 3:

                        MainMenu();

*break*;

*default*:

                        printf(" Wrong Input! Press any key to try again...");

                        clear(); getchar();

*goto* Copy\_Menu;

                }

*break*;

*case* 5:

                Merge\_Menu:

                system("cls"); *//Sub menu for merge*

                printf("     Merge Menu\n-----------------------\n  Select an option.\n-----------------------");

                printf("\n1) Merge Array 1 and Array 2 to Array 3");

                printf("\n2) Go back.");

                printf("\n\nInput: ");

                clear();

                scanf("%d", &input);

*switch* (input)

                {

*case* 1:

                        system("cls");

                        Merge(arrayA, countA, arrayB, countB, arrayC, &countC);

                        Display(arrayC, countC, 'C');

                        MainMenu();

*break*;

*case* 2:

                        MainMenu();

*break*;

*default*:

                        printf("Wrong Input! Press any key to try again...");

                        clear(); getchar();

                        system("cls");

*goto* Merge\_Menu;

                }

*case* 6:

                exit(0);

*default*:

                printf(" Wrong Input! Press any key to try again...");

                clear(); getchar();

                MainMenu();

    }

}

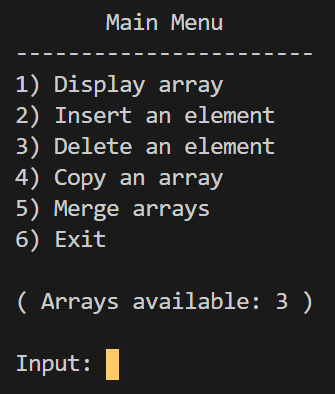
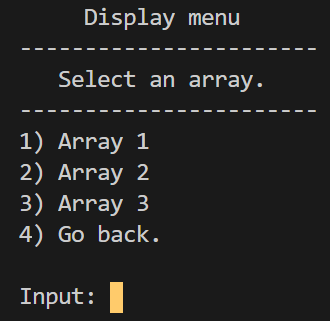
int main()

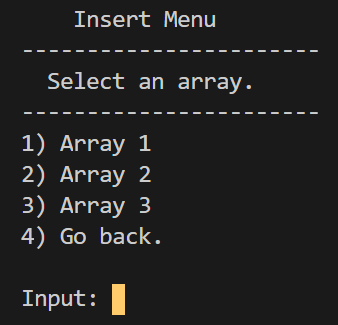
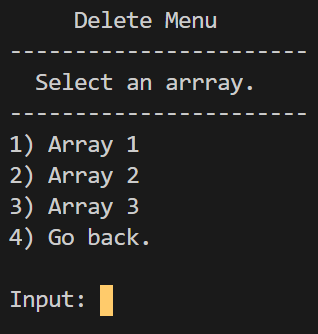
{

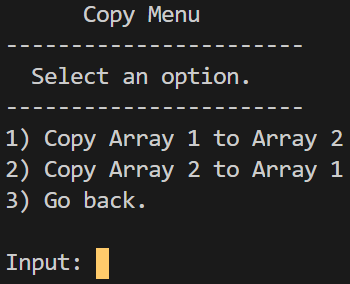
    MainMenu();

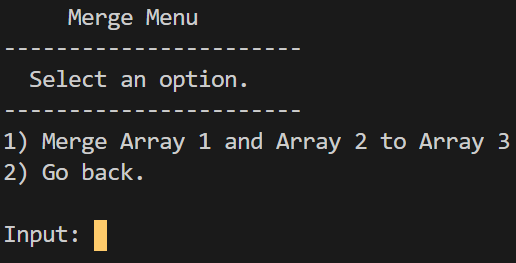
}

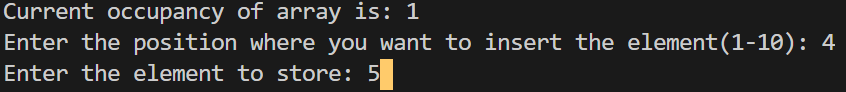
**Screenshot of compilation and execution:**

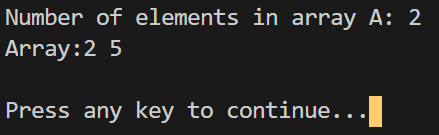
 









1. **Activity:** Convert uppercase string to lowercase string using loop.

**Algorithm:**

1. Get string input from user.
2. Iterate through all letters, checking each if it is an uppercase letter
3. If a letter is uppercase, increase its ASCII value by 32 making it lowercase.
4. Print the converted string.\

**Test Case Table:**

|  |  |  |
| --- | --- | --- |
| Entered String | Printed String | Pass/Fail |
| HELLO | hello | Pass |
| Hi how ARE you | hi how are you | Pass |
| This is a tEsT CASE | this is a test case | Pass |

**Program:**

*#include* <stdio.h>

void main(){

    char s[100];

    printf("Enter string: ");

*//Get user input*

    scanf("%[^\n]%\*c", s);

*for* (int i = 0; s[i]; i++)

    {

*//Chack if letter in check is uppercase*

*if* (s[i] >= 65 && s[i] <= 90)

        {

*//If uppercase, increase ASCII value by 32*

            s[i] += 32;

        }

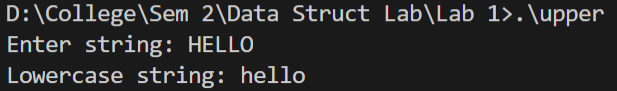
    }

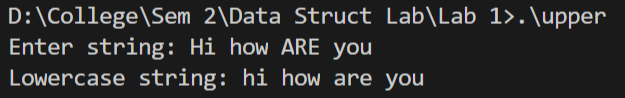
*//Print*

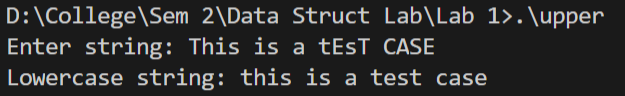
    printf("Lowercase string: %s", s);

}

**Screenshot of compilation and execution:**

****

****

****

1. **Activity:** Find product of 2 matrices using pointers.

**Algorithm:**

1. Get input: matrix sizes
2. Check whether the two matrix can be multiplied.
3. Create 3 dynamic arrays for the matrices 1, 2 and result
4. Get entries for matrices 1 and 2
5. Pass the matrices to matrix multiply function via pass by reference:
   1. Create 2 for loops to iterate all elements through the resultant array
   2. Set each element in result matrix as the sum of product of element in row of matrix 1 and column of matrix 2 (General matrix multiplication)
   3. Exit out of the function
6. Create double for loop to print out the resultant matrix.

**Test Case Table:**

|  |  |  |  |
| --- | --- | --- | --- |
| Matrix 1 | Matrix 2 | Result | Pass/Fail |
| 1 2 3  4 5 6  7 8 9 | 1 2 3  4 5 6  7 8 9 | 30 36 42  66 81 96  102 126 150 | Pass |
| 1 2 3  4 5 6 | 10 11  20 21  30 31 | 140 146  320 335 | Pass |

**Program:**

*#include*<stdio.h>

*#include* <stdlib.h>

*//Defining 3x3 matrix*

*#define* ROW 3

*#define* COL 3

*//Get matrix data*

void matrixInput(int \*mat, int n , int m)

{

    int row, col;

*for* (row = 0; row < n; row++)

    {

*for* (col = 0; col < m; col++)

        {

            scanf("%d", (mat + row\*m + col));

        }

    }

}

*//Function to multiply Matrices mat1 \* mat2 = res*

void matrixMultiply(int \*mat1, int \*mat2, int \*res, int n1, int n2, int m1, int m2)

{

    int row, col, i;

    int sum;

*for* (row = 0; row < n1; row++)

    {

*for* (col = 0; col < m2; col++)

        {

            sum = 0;

*for* (int e = 0; e < m1; e++)

            {

                sum += (\*(mat1 + row\*m1 + e))\*(\*(mat2 + col + e\*(n2-1)));

            }

            \*(res + row\*m2 + col) = sum;

        }

    }

}

void main()

{

    system("cls");

    L1: printf("Enter dimentsions of matrix 1: ");

    int n1, m1;

    scanf("%d %d", &n1, &m1);

    printf("Enter dimentsions of matrix 2: ");

    int n2, m2;

    scanf("%d %d", &n2, &m2);

*//check if multiplication is possible, 1st matrix rows must be equal to*

*//Second matrix columns*

*if* (m1 != n2)

    {

        system("cls");

        printf("Matrices can not be multiplied.\nTry again..\n");

*goto* L1;

    }

    int \*mat1 = (int \*)malloc(n1\*m1\*sizeof(int));

    int \*mat2 = (int \*)malloc(n2\*m2\*sizeof(int));

    int \*res = (int \*)malloc(n1\*m2\*sizeof(int));

    printf("Enter First matrix\n");

    matrixInput(mat1, n1, m1);

    printf("Enter Second matrix\n");

    matrixInput(mat2, n2, m2);

    matrixMultiply(mat1, mat2, res, n1, n2, m1, m2);

*//print the resultant matrices*

    printf("Resultant Matrix\n");

*for* (int i = 0; i < n1; i++)

    {

*for* (int j = 0; j < m2; j++)

        {

            printf("%d\t", \*(res + i\*m2 + j));

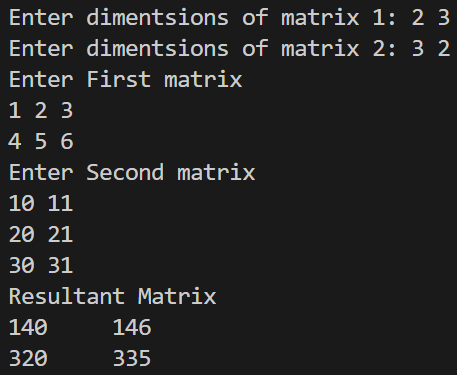
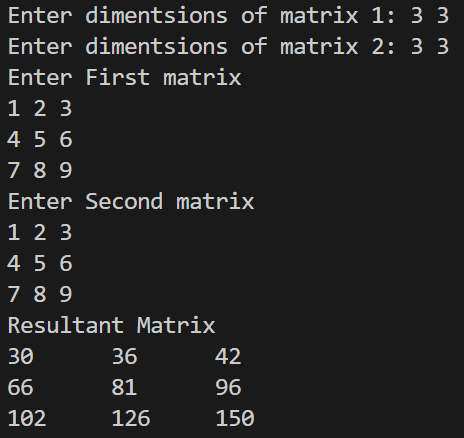
        }

        printf("\n");

    }

}

**Screenshot of compilation and execution:**

 ****