**ASSIGNMENT – 2**

1. Write a C program to read a 2D array (with most of the elements as 0s) and then represent the same array as Sparse Metrics.

#include <stdio.h>

#define max\_row 10

#define max\_col 10

#define max\_element max\_row \*max\_col

void SparseMatrix(int arr[max\_row][max\_col], int rows, int cols)

{

int sparse[max\_element][3], k = 0, i, j;

for (i = 0; i < rows; ++i)

{

for (j = 0; j < cols; ++j)

{

if (arr[i][j] != 0)

{

sparse[k][0] = i;

sparse[k][1] = j;

sparse[k][2] = arr[i][j];

k++;

}

}

}

printf("Sparse Matrix:\n");

for (i = 0; i < k; ++i)

{

for (j = 0; j < 3; ++j)

{

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

}

printf("\n");

}

}

void main()

{

int rows, cols, i, j;

printf("Enter Number of Rows: ");

scanf("%d", &rows);

printf("Enter Number of Columns: ");

scanf("%d", &cols);

int array[max\_row][max\_col];

printf("Enter Matrix Elements:\n");

for (i = 0; i < rows; ++i)

{

for (j = 0; j < cols; ++j)

{

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

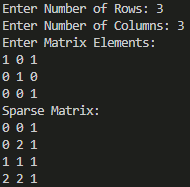
}

}

SparseMatrix(array, rows, cols);

}

Output:



1. Write a C program to pass an array to a function using Call by Value, update the array values in the function, print the array elements both in the function and in the calling function.

#include <stdio.h>

void updateAndPrint(int arr[], int n)

{

int i;

printf("\nUpdating array elements in the function:\n");

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

{

arr[i] \*= 2;

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

}

printf("\n");

}

void main()

{

int n, i;

printf("Enter the size of the array : ");

scanf("%d", &n);

int arr[100];

printf("Enter %d elements for the array:\n", n);

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

{

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

}

printf("Array elements before calling the function:\n");

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

{

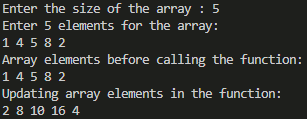
printf("%d ", arr[i]);

}

updateAndPrint(arr, n);

}

Output:



1. Write a C program to pass an array to a function using Call by Reference, update the array values in the function, print the array elements both in the function and in the calling function.

#include <stdio.h>

void updateAndPrint(int \*arr, int n)

{

int \*ptr = &arr[0];

int i;

printf("\nUpdating array elements in the function:\n");

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

{

\*ptr \*= 2;

printf("%d ", \*ptr);

ptr += 1;

}

printf("\n");

}

int main()

{

int n, i;

printf("Enter the size of the array : ");

scanf("%d", &n);

int arr[100];

printf("Enter %d elements for the array:\n", n);

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

{

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

}

printf("\nArray elements before calling the function:\n");

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

{

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

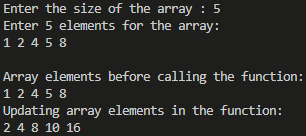
}

updateAndPrint(&arr[0], n);

return 0;

}

Output:



1. Write a program to display n number of elements. Memory should be allocated dynamically using malloc().

#include <stdio.h>

#include <stdlib.h>

int main()

{

int n, i;

printf("Enter the number of elements: ");

scanf("%d", &n);

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

if (arr == NULL)

{

printf("Memory allocation failed.\n");

return 1;

}

printf("Enter %d elements:\n", n);

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

{

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

}

printf("\nElements of the array:\n");

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

{

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

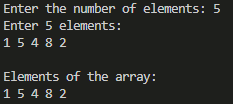
}

free(arr);

return 0;

}

Output:



1. Write a program to display n number of elements. Memory should be allocated dynamically using calloc().

#include <stdio.h>

#include <stdlib.h>

int main()

{

int n, i;

printf("Enter the number of elements: ");

scanf("%d", &n);

int \*arr = (int \*)calloc(n, sizeof(int));

if (arr == NULL)

{

printf("Memory allocation failed.\n");

return 1;

}

printf("Enter %d elements:\n", n);

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

{

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

}

printf("\nElements of the array:\n");

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

{

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

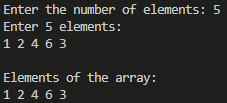
}

free(arr);

return 0;

}

Output:



1. Write a program to allocate memory using malloc() and then reallocate the previously allocated memory using realloc(). Display the elements which have been taken after reallocation.

#include <stdio.h>

#include <stdlib.h>

int main()

{

int n, new\_size, i;

printf("Enter the number of elements: ");

scanf("%d", &n);

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

if (arr == NULL)

{

printf("Memory allocation failed.\n");

return 1;

}

printf("Enter %d elements:\n", n);

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

{

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

}

printf("\nElements before reallocation:\n");

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

{

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

}

printf("\nEnter the new size for reallocation: ");

scanf("%d", &new\_size);

arr = (int \*)realloc(arr, new\_size \* sizeof(int));

if (arr == NULL)

{

printf("Memory reallocation failed.\n");

return 1;

}

if (new\_size > n)

{

printf("Enter %d additional elements:\n", new\_size - n);

for (i = n; i < new\_size; ++i)

{

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

}

}

printf("\nElements after reallocation:\n");

for (i = 0; i < new\_size; ++i)

{

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

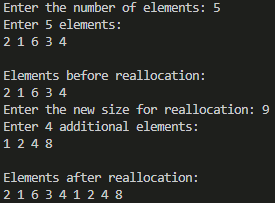
}

free(arr);

return 0;

}

Output:



1. Write a program to allocate memory using calloc() and then reallocate the previously allocated memory using realloc(). Display the elements which have been taken after reallocation.

#include <stdio.h>

#include <stdlib.h>

int main()

{

int n, new\_size, i;

printf("Enter the number of elements: ");

scanf("%d", &n);

int \*arr = (int \*)calloc(n, sizeof(int));

if (arr == NULL)

{

printf("Memory allocation failed.\n");

return 1;

}

printf("Enter %d elements:\n", n);

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

{

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

}

printf("\nElements before reallocation:\n");

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

{

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

}

printf("\nEnter the new size for reallocation: ");

scanf("%d", &new\_size);

if (new\_size > n)

{

arr = (int \*)realloc(arr, new\_size \* sizeof(int));

if (arr == NULL)

{

printf("Memory reallocation failed.\n");

return 1;

}

printf("Enter %d additional elements:\n", new\_size - n);

for (i = n; i < new\_size; ++i)

{

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

}

}

printf("\nElements after reallocation:\n");

for (i = 0; i < new\_size; ++i)

{

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

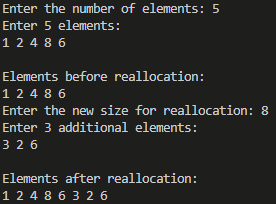
}

free(arr);

return 0;

}

Output:



1. Write a C program to search an element in an Array using dynamic memory allocation.

#include <stdio.h>

#include <stdlib.h>

int searchElement(int \*arr, int size, int key)

{

int i;

for (i = 0; i < size; ++i)

{

if (arr[i] == key)

{

return i;

}

}

return -1;

}

int main()

{

int n, key, i;

printf("Enter the size of the array: ");

scanf("%d", &n);

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

if (arr == NULL)

{

printf("Memory allocation failed.\n");

return 1;

}

printf("Enter %d elements for the array:\n", n);

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

{

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

}

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

scanf("%d", &key);

int index = searchElement(arr, n, key);

if (index != -1)

{

printf("Element %d found at index %d.\n", key, index);

}

else

{

printf("Element %d not found in the array.\n", key);

}

free(arr);

return 0;

}

Output:

