**Saksham 056 BCA 2A**

**Index**

|  |  |  |  |
| --- | --- | --- | --- |
| **P.No.** | **Title** | **Date (2A)** | **Date (2B)** |
|  | **Arrays** |  |  |
| 1 | Write a program to  (a.) create an array of integers and initialize it at compile-time (b.) create another array of floating values and initialize it at run-time (c.) display the elements of both the arrays with proper headings | 20-Jan-25 | 23-Jan-25 |
| 2 | Write a Program to implement Linear Search for  (a.) First occurrence of search item (b.) All occurrences of search item | 20-Jan-25 | 23-Jan-25 |
| 3 | Write a program to  (a.) create an array of integers and initialize it (b.) Find minimum and maximum elements in the array (c.) Find sum and average of array elements | 20-Jan-25 | 23-Jan-25 |
| 4 | Write a program to Merge unsorted arrays | 20-Jan-25 | 23-Jan-25 |
| 5 | Write a program to Marge sorted arrays | 20-Jan-25 | 23-Jan-25 |
| 6 | Write a program to insert a number at a given location in an array. | 27-Jan-25 | 30-Jan-25 |
| 7 | Write a program to delete a number from a given location in an array. | 27-Jan-25 | 30-Jan-25 |
| 8 | Write a program to search a number in an array and delete it, if found. | 27-Jan-25 | 30-Jan-25 |
| 9 | Write a Program to implement Binary Search | 27-Jan-25 | 30-Jan-25 |
| 10 | Write a Program to perform elimination of duplicate elements from an existing list of elements. | 27-Jan-25 | 30-Jan-25 |
|  | **2-D Arrays and Sparse matrix** |  |  |
| 11 | Create a Matrix. Perform addition, subtraction, Transpose and Multiplication using Switch-Case statement. | 03-Feb-25 | 06-Feb-25 |
| 12 | Read and display Upper, Lower and Tri-diagonal matrices | 03-Feb-25 | 06-Feb-25 |
| 13 | Implement sparse matrices using 3-tuple notation. | 03-Feb-25 | 06-Feb-25 |
|  | **Sorting methods** |  |  |
| 14 | Write a Program to implement Selection Sort. | 03-Feb-25 | 06-Feb-25 |

Question 1:

Write a program to   
(a.) create an array of integers and initialize it at compile-time  
(b.) create another array of floating values and initialize it at run-time  
(c.) display the elements of both the arrays with proper headings

Source Code:

|  |
| --- |
| /\*"Write a program to  (a.) create an array of integers and initialize it at compile-time  (b.) create another array of floating values and initialize it at run-time  (c.) display the elements of both the arrays with proper headings"\*/  #include <stdio.h>  int main() {  // (a) Create and initialize an array of integers at compile-time  int intArray[] = {10, 20, 30, 40, 50};  int intArraySize = sizeof(intArray) / sizeof(intArray[0]);  // (b) Create another array of floating values and initialize it at run-time  float floatArray[5]; // Array for 5 floating-point numbers  int floatArraySize = sizeof(floatArray) / sizeof(floatArray[0]);  printf("\nEnter %d floating-point values:\n", floatArraySize);  for (int i = 0; i < floatArraySize; i++) {  scanf("%f", &floatArray[i]);  }  // (c) Display both arrays with proper headings  printf("\nInteger Array (Compile-time initialized):\n");  for (int i = 0; i < intArraySize; i++) {  printf("%d ", intArray[i]);  }  printf("\n");  printf("\nFloating-point Array (Run-time initialized):\n");  for (int i = 0; i < floatArraySize; i++) {  printf("%.2f ", floatArray[i]);  }  printf("\n");  return 0;  } |

Output:

|  |
| --- |
| Enter 5 floating-point values:  5.05  4.05  3.023  55.43  43453.33  Integer Array (Compile-time initialized):  10 20 30 40 50  Floating-point Array (Run-time initialized):  5.05 4.05 3.02 55.43 43453.33 |

Question 2:

Write a Program to implement Linear Search for   
(a.) First occurrence of search item  
(b.) All occurrences of search items

Source Code:

|  |  |
| --- | --- |
| /\*"Write a Program to implement Linear Search for   (a.) First occurence of search item   (b.) All occurences of search item"\*/  #include <stdio.h>  #include <stdlib.h>  // Required for malloc and free  void searchFirstOccurrence(int arr[], int size, int key) {      for (int i = 0; i < size; i++) {          if (arr[i] == key) {              printf("First occurrence of %d found at index %d\n", key, i);              return;          }      }      printf("Element %d not found in the array.\n", key);  }  void searchAllOccurrences(int arr[], int size, int key) {      int found = 0;      printf("All occurrences of %d found at indices: ", key);      for (int i = 0; i < size; i++) {          if (arr[i] == key) {              printf("%d ", i);              found = 1;          }      }      if (!found) {          printf("Element %d not found in the array.", key);      }      printf("\n");  } | int main() {      int size, key;      // Input array size      printf("\nEnter the size of the array: ");      scanf("%d", &size);      // Dynamic memory allocation      int \*arr = (int \*)malloc(size \* sizeof(int));      if (arr == NULL) {          printf("Memory allocation failed!\n");          return 1;      }      // Input array elements      printf("Enter %d elements:\n", size);      for (int i = 0; i < size; i++) {          scanf("%d", &arr[i]);      }      // Input key to search      printf("Enter the element to search: ");      scanf("%d", &key);      // (a) Find first occurrence      searchFirstOccurrence(arr, size, key);      // (b) Find all occurrences      searchAllOccurrences(arr, size, key);      // Free allocated memory      free(arr);      return 0;  } |

Output:

|  |
| --- |
| Enter the size of the array: 5  Enter 5 elements:  1  2  3  3  4  Enter the element to search: 3  First occurrence of 3 found at index 2  All occurrences of 3 found at indices: 2 3 |

Question 3:

Write a program to Merge unsorted arrays

Source Code:

|  |  |
| --- | --- |
| /\*"Write a program to  (a.) create an array of integers and initialize it  (b.) Find minimum and maximum elements in the array  (c.) Find sum and average of array elements"\*/  #include <stdio.h>  #include <stdlib.h> // For malloc() and free()  // Function to find the minimum element in the array  int findMin(int arr[]) {  int size = 0;  while (arr[size] != '\0') size++; // Calculate size dynamically  int min = arr[0]; // Initialize min with the first element  for (int i = 1; i < size; i++) {  if (arr[i] < min) {  min = arr[i];  }  }  return min;  }  // Function to find the maximum element in the array  int findMax(int arr[]) {  int size = 0;  while (arr[size] != '\0') size++; // Calculate size dynamically  int max = arr[0]; // Initialize max with the first element  for (int i = 1; i < size; i++) {  if (arr[i] > max) {  max = arr[i];  }  }  return max;  }  // Function to calculate the sum of array elements  int calculateSum(int arr[]) {  int size = 0, sum = 0;  while (arr[size] != '\0') { // Calculate size dynamically  sum += arr[size];  size++;  }  return sum;  } | // Function to calculate the average by calling calculateSum  float calculateAverage(int arr[]) {  int size = 0;  while (arr[size] != '\0') size++; // Calculate size dynamically  int sum = calculateSum(arr); // Call sum function  return (float)sum / size;  }  int main() {  int size;  // (a) Input array size  printf("\nEnter the size of the array: ");  scanf("%d", &size);  // Dynamic memory allocation  int \*arr = (int \*)malloc((size + 1) \* sizeof(int)); // +1 to add NULL termination  if (arr == NULL) {  printf("Memory allocation failed!\n");  return 1;  }  // Input array elements  printf("Enter %d elements:\n", size);  for (int i = 0; i < size; i++) {  scanf("%d", &arr[i]);  }  arr[size] = '\0'; // Null termination for size calculation  // (b) Find minimum and maximum elements  int min = findMin(arr);  int max = findMax(arr);  // (c) Calculate sum and average  int sum = calculateSum(arr);  float average = calculateAverage(arr);  // Display results  printf("\nMinimum element: %d\n", min);  printf("Maximum element: %d\n", max);  printf("Sum of elements: %d\n", sum);  printf("Average of elements: %.2f\n", average);  // Free allocated memory  free(arr);  return 0;  } |

Output:

|  |
| --- |
| Enter the size of the array: 6  Enter 6 elements: 1 1 2 3 4 4  Minimum element: 1  Maximum element: 4  Sum of elements: 15  Average of elements: 2.50 |

Question 4:

Write a program to Merge unsorted arrays

Source Code:

|  |  |
| --- | --- |
| // Write a program to Merge unsorted arrays  #include <stdio.h>  #include <stdlib.h> // For malloc() and free()  // Function to merge two arrays  void mergeArrays(int \*arr1, int size1, int \*arr2, int size2, int \*mergedArr) {  int i, j;    // Copy elements of arr1 to mergedArr  for (i = 0; i < size1; i++) {  mergedArr[i] = arr1[i];  }  // Copy elements of arr2 to mergedArr  for (j = 0; j < size2; j++) {  mergedArr[i + j] = arr2[j];  }  }  int main() {  int size1, size2;  // Input size of first array  printf("Enter size of first array: ");  scanf("%d", &size1);  // Dynamic memory allocation for first array  int \*arr1 = (int \*)malloc(size1 \* sizeof(int));  if (arr1 == NULL) {  printf("Memory allocation failed!\n");  return 1;  }  // Input elements of first array  printf("Enter %d elements of first array:\n", size1);  for (int i = 0; i < size1; i++) {  scanf("%d", &arr1[i]);  }  // Input size of second array  printf("Enter size of second array: ");  scanf("%d", &size2); | // Dynamic memory allocation for second array  int \*arr2 = (int \*)malloc(size2 \* sizeof(int));  if (arr2 == NULL) {  printf("Memory allocation failed!\n");  free(arr1); // Free previously allocated memory before exiting  return 1;  }  // Input elements of second array  printf("Enter %d elements of second array:\n", size2);  for (int i = 0; i < size2; i++) {  scanf("%d", &arr2[i]);  }  // Allocate memory for merged array  int \*mergedArr = (int \*)malloc((size1 + size2) \* sizeof(int));  if (mergedArr == NULL) {  printf("Memory allocation failed!\n");  free(arr1);  free(arr2);  return 1;  }  // Merge the two arrays  mergeArrays(arr1, size1, arr2, size2, mergedArr);  // Display merged array  printf("\nMerged Array:\n");  for (int i = 0; i < size1 + size2; i++) {  printf("%d ", mergedArr[i]);  }  printf("\n");  // Free allocated memory  free(arr1);  free(arr2);  free(mergedArr);  return 0;  } |

Output:

|  |
| --- |
| Enter size of first array: 5  Enter 5 elements of first array: 1 2 3 4 2  Enter size of second array: 6  Enter 6 elements of second array: 4 2 3 4 2 4  Merged Array:  1 2 3 4 2 4 2 3 4 2 4 |

Question 5:

Write a program to Merge Sorted arrays

Source Code:

|  |  |
| --- | --- |
| // Write a program to Marge sorted arrays  #include <stdio.h>  #include <stdlib.h> // For malloc() and free()  // Function to merge two sorted arrays into a single sorted array  void mergeSortedArrays(int \*arr1, int size1, int \*arr2, int size2, int \*mergedArr) {  int i = 0, j = 0, k = 0;  // Merge elements in sorted order  while (i < size1 && j < size2) {  if (arr1[i] <= arr2[j]) {  mergedArr[k++] = arr1[i++];  } else {  mergedArr[k++] = arr2[j++];  }  }  // Copy remaining elements of arr1 (if any)  while (i < size1) {  mergedArr[k++] = arr1[i++];  }  // Copy remaining elements of arr2 (if any)  while (j < size2) {  mergedArr[k++] = arr2[j++];  }  }  int main() {  int size1, size2;  // Input size of first array  printf("Enter size of first sorted array: ");  scanf("%d", &size1);  // Dynamic memory allocation for first array  int \*arr1 = (int \*)malloc(size1 \* sizeof(int));  if (arr1 == NULL) {  printf("Memory allocation failed!\n");  return 1;  }  // Input elements of first sorted array  printf("Enter %d elements (in sorted order) for first array:\n", size1);  for (int i = 0; i < size1; i++) {  scanf("%d", &arr1[i]);  } | // Input size of second array  printf("Enter size of second sorted array: ");  scanf("%d", &size2);  // Dynamic memory allocation for second array  int \*arr2 = (int \*)malloc(size2 \* sizeof(int));  if (arr2 == NULL) {  printf("Memory allocation failed!\n");  free(arr1);  return 1;  }  // Input elements of second sorted array  printf("Enter %d elements (in sorted order) for second array:\n", size2);  for (int i = 0; i < size2; i++) {  scanf("%d", &arr2[i]);  }  // Allocate memory for merged array  int \*mergedArr = (int \*)malloc((size1 + size2) \* sizeof(int));  if (mergedArr == NULL) {  printf("Memory allocation failed!\n");  free(arr1);  free(arr2);  return 1;  }  // Merge the two sorted arrays  mergeSortedArrays(arr1, size1, arr2, size2, mergedArr);  // Display merged sorted array  printf("\nMerged Sorted Array:\n");  for (int i = 0; i < size1 + size2; i++) {  printf("%d ", mergedArr[i]);  }  printf("\n");  // Free allocated memory  free(arr1);  free(arr2);  free(mergedArr);  return 0;  } |

Output:

|  |
| --- |
| Enter size of first sorted array: 6  Enter 6 elements (in sorted order) for first array: 1 3 4 6 8 10  Enter size of second sorted array: 2  Enter 2 elements (in sorted order) for second array: 5 7  Merged Sorted Array:  1 3 4 5 6 7 8 10 |

Question 6:

Write a program to insert a number at a given location in an array.

Source Code:

|  |  |
| --- | --- |
| // Write a program to insert a number at a given location in an array.  #include <stdio.h>  #define MAX\_SIZE 100 // Define a maximum size for safety  int arrLen;  void arrPrint(int arr[]) {  printf("\n");  for (int i = 0; i < arrLen; i++) {  printf("%d ", arr[i]);  }  printf("\n");  }  void arrInsert(int arr[], int num, int pos) {  if (pos < 1 || pos > arrLen + 1) {  printf("Invalid position!\n");  return;  }  if (arrLen >= MAX\_SIZE) {  printf("Array is full! Cannot insert.\n");  return;  }  for (int i = arrLen; i >= pos; i--) {  arr[i] = arr[i - 1];  }  arr[pos - 1] = num;  arrLen++;  printf("Updated Array: ");  arrPrint(arr);  } | int main() {  int arr[MAX\_SIZE], num, pos;  printf("Enter Number of Elements in Array: ");  scanf("%d", &arrLen);  for (int i = 0; i < arrLen; i++) {  printf("Enter Element %d: ", i + 1);  scanf("%d", &arr[i]);  }  printf("Enter element to insert: ");  scanf("%d", &num);  printf("Enter position: ");  scanf("%d", &pos);  arrInsert(arr, num, pos);    return 0;  } |

Output:

|  |
| --- |
| Enter Number of Elements in Array: 6  Enter Element 1: 1  Enter Element 2: 2  Enter Element 3: 3  Enter Element 4: 4  Enter Element 5: 5  Enter Element 6: 6  Enter element to insert: 9  Enter position: 4  Updated Array:  1 2 3 9 4 5 6 |

Question 7:

Write a program to delete a number from a given location in an array.

Source Code:

|  |  |
| --- | --- |
| // Write a program to delete a number from a given location in an array.  #include <stdio.h>  #define MAX\_SIZE 100 // Define a maximum size for safety  int arrLen;  void arrPrint(int arr[]) {  printf("\n");  for (int i = 0; i < arrLen; i++) {  printf("%d ", arr[i]);  }  printf("\n");  }  void arrDelete(int arr[], int pos) {  if (pos < 1 || pos > arrLen) {  printf("Invalid position!\n");  return;  }  for (int i = pos - 1; i < arrLen - 1; i++) {  arr[i] = arr[i + 1];  }  arrLen--;  printf("Updated Array: ");  arrPrint(arr);  } | int main() {  int arr[MAX\_SIZE], pos;  printf("Enter Number of Elements in Array: ");  scanf("%d", &arrLen);  for (int i = 0; i < arrLen; i++) {  printf("Enter Element %d: ", i + 1);  scanf("%d", &arr[i]);  }  printf("Enter position to delete: ");  scanf("%d", &pos);  arrDelete(arr, pos);  return 0;  } |

Output:

|  |
| --- |
| Enter Number of Elements in Array: 6  Enter Element 1: 1  Enter Element 2: 2  Enter Element 3: 4  Enter Element 4: 5  Enter Element 5: 6  Enter Element 6: 7  Enter position to delete: 4  Updated Array:  1 2 4 6 7 |

Question 8:

Write a program to search a number in an array and delete it, if found.

Source Code:

|  |  |
| --- | --- |
| // Write a program to search a number in an array and delete it, if found.  #include <stdio.h>  #define MAX\_SIZE 100 // Define a maximum size for safety  int arrLen;  void arrPrint(int arr[]) {  printf("\n");  for (int i = 0; i < arrLen; i++) {  printf("%d ", arr[i]);  }  printf("\n");  }  void arrDelete(int arr[], int pos) {  for (int i = pos - 1; i < arrLen - 1; i++) {  arr[i] = arr[i + 1];  }  arrLen--;  }  void arrSearchDel(int arr[], int num) {  int pos = -1;  for (int i = 0; i < arrLen; i++) {  if (arr[i] == num) {  pos = i + 1;  break;  }  }  if (pos == -1) {  printf("Element not found!\n");  return;  }  arrDelete(arr, pos);  printf("Updated Array: ");  arrPrint(arr);  } | int main() {  int arr[MAX\_SIZE], num;  printf("Enter Number of Elements in Array: ");  scanf("%d", &arrLen);  for (int i = 0; i < arrLen; i++) {  printf("Enter Element %d: ", i + 1);  scanf("%d", &arr[i]);  }  printf("Enter element to search and delete: ");  scanf("%d", &num);  arrSearchDel(arr, num);  return 0;  } |

Output:

|  |
| --- |
| Enter Number of Elements in Array: 7  Enter Element 1: 1  Enter Element 2: 2  Enter Element 3: 3  Enter Element 4: 4  Enter Element 5: 3  Enter Element 6: 2  Enter Element 7: 1  Enter element to search and delete: 4  Updated Array:  1 2 3 3 2 1 |

Question 9:

Write a Program to implement Binary Search

Source Code:

|  |  |
| --- | --- |
| // Write a Program to implement Binary Search  #include <stdio.h>  #define MAX\_SIZE 100 // Define a maximum size for safety  int arrLen;  void arrPrint(int arr[]) {  printf("\n");  for (int i = 0; i < arrLen; i++) {  printf("%d ", arr[i]);  }  printf("\n");  }  void arrSort(int arr[]) {  for (int i = 0; i < arrLen - 1; i++) {  for (int j = i + 1; j < arrLen; j++) {  if (arr[i] > arr[j]) {  int temp = arr[i];  arr[i] = arr[j];  arr[j] = temp;  }  }  }  printf("Sorted Array: ");  arrPrint(arr);  }  void arrBiSearch(int arr[], int num) {  int low = 0, high = arrLen - 1, mid;  while (low <= high) {  mid = (low + high) / 2;  if (arr[mid] == num) {  printf("Element found at Position: %d\n", mid + 1);  return;  } else if (arr[mid] < num) {  low = mid + 1;  } else {  high = mid - 1;  }  }  printf("Element not found!\n");  } | int main() {  int arr[MAX\_SIZE], num;  printf("Enter Number of Elements in Array: ");  scanf("%d", &arrLen);  for (int i = 0; i < arrLen; i++) {  printf("Enter Element %d: ", i + 1);  scanf("%d", &arr[i]);  }  arrSort(arr);  printf("Enter element to search: ");  scanf("%d", &num);  arrBiSearch(arr, num);  return 0;  } |

Output:

|  |
| --- |
| Enter Number of Elements in Array: 7  Enter Element 1: 9  Enter Element 2: 5  Enter Element 3: 4  Enter Element 4: 7  Enter Element 5: 4  Enter Element 6: 5  Enter Element 7: 7  Sorted Array: 4 4 5 5 7 7 9  Enter element to search: 9  Element found at Position: 7 |

Question 10:

Write a Program to perform elimination of duplicate elements from an existing list of elements.

Source Code:

|  |  |
| --- | --- |
| // Write a Program to perform elimination of duplicate elements from an existing list of elements.  #include <stdio.h>  #define MAX\_SIZE 100 // Define a maximum size for safety  int arrLen;  void arrPrint(int arr[]) {  printf("\n");  for (int i = 0; i < arrLen; i++) {  printf("%d ", arr[i]);  }  printf("\n");  }  void arrDelete(int arr[], int pos) {  for (int i = pos - 1; i < arrLen - 1; i++) {  arr[i] = arr[i + 1];  }  arrLen--;  }  void arrElmDup(int arr[]) {  for (int i = 0; i < arrLen; i++) {  for (int j = i + 1; j < arrLen; ) {  if (arr[i] == arr[j]) {  arrDelete(arr, j + 1);  } else {  j++; // Only increment when no deletion occurs  }  }  }  printf("Updated Array: ");  arrPrint(arr);  } | int main() {  int arr[MAX\_SIZE];  printf("Enter Number of Elements in Array: ");  scanf("%d", &arrLen);  for (int i = 0; i < arrLen; i++) {  printf("Enter Element %d: ", i + 1);  scanf("%d", &arr[i]);  }  arrElmDup(arr);  return 0;  } |

Output:

|  |
| --- |
| Enter Number of Elements in Array: 10  Enter Element 1: 4  Enter Element 2: 4  Enter Element 3: 5  Enter Element 4: 7  Enter Element 5: 5  Enter Element 6: 2  Enter Element 7: 6  Enter Element 8: 8  Enter Element 9: 4  Enter Element 10: 79  Updated Array:  4 5 7 2 6 8 79 |

Question 11:

Create a Matrix. Perform addition, subtraction, Transpose and Multiplication using Switch-Case statement.

Source Code:

|  |  |
| --- | --- |
| // C Program for Matrix Operations Using Switch-Case  #include <stdio.h>  #define SIZE 3 // Define matrix size  void inputMatrix(int matrix[SIZE][SIZE], char name) {  printf("Enter elements of matrix %c (%dx%d):\n", name, SIZE, SIZE);  for (int i = 0; i < SIZE; i++) {  for (int j = 0; j < SIZE; j++) {  printf("%c[%d][%d]: ", name, i, j);  scanf("%d", &matrix[i][j]);  }  }  }  void printMatrix(int matrix[SIZE][SIZE]) {  for (int i = 0; i < SIZE; i++) {  for (int j = 0; j < SIZE; j++) {  printf("%d\t", matrix[i][j]);  }  printf("\n");  }  }  void addMatrices(int A[SIZE][SIZE], int B[SIZE][SIZE], int result[SIZE][SIZE]) {  for (int i = 0; i < SIZE; i++) {  for (int j = 0; j < SIZE; j++) {  result[i][j] = A[i][j] + B[i][j];  }  }  }  void subtractMatrices(int A[SIZE][SIZE], int B[SIZE][SIZE], int result[SIZE][SIZE]) {  for (int i = 0; i < SIZE; i++) {  for (int j = 0; j < SIZE; j++) {  result[i][j] = A[i][j] - B[i][j];  }  }  }  void multiplyMatrices(int A[SIZE][SIZE], int B[SIZE][SIZE], int result[SIZE][SIZE]) {  for (int i = 0; i < SIZE; i++) {  for (int j = 0; j < SIZE; j++) {  result[i][j] = 0;  for (int k = 0; k < SIZE; k++) {  result[i][j] += A[i][k] \* B[k][j];  }  }  }  }  void transposeMatrix(int A[SIZE][SIZE], int result[SIZE][SIZE]) {  for (int i = 0; i < SIZE; i++) {  for (int j = 0; j < SIZE; j++) {  result[j][i] = A[i][j];  }  }  } | int main() {  int A[SIZE][SIZE], B[SIZE][SIZE], result[SIZE][SIZE];  int choice;  // Input matrices  inputMatrix(A, 'A');  inputMatrix(B, 'B');  // Menu  printf("\nChoose operation:\n");  printf("1. Addition\n2. Subtraction\n3. Multiplication\n4. Transpose (of A)\n");  printf("Enter your choice: ");  scanf("%d", &choice);  switch (choice) {  case 1:  addMatrices(A, B, result);  printf("\nResultant Matrix after Addition:\n");  printMatrix(result);  break;  case 2:  subtractMatrices(A, B, result);  printf("\nResultant Matrix after Subtraction:\n");  printMatrix(result);  break;  case 3:  multiplyMatrices(A, B, result);  printf("\nResultant Matrix after Multiplication:\n");  printMatrix(result);  break;  case 4:  transposeMatrix(A, result);  printf("\nTranspose of Matrix A:\n");  printMatrix(result);  break;  default:  printf("\nInvalid choice!\n");  }  return 0;  } |

Output:

|  |  |
| --- | --- |
| Enter elements of matrix A (3x3):  A[0][0]: 1  A[0][1]: 2  A[0][2]: 3  A[1][0]: 4  A[1][1]: 5  A[1][2]: 6  A[2][0]: 7  A[2][1]: 8  A[2][2]: 9  Enter elements of matrix B (3x3):  B[0][0]: 9  B[0][1]: 8  B[0][2]: 8  B[1][0]: 7  B[1][1]: 6  B[1][2]: 5  B[2][0]: 4  B[2][1]: 3  B[2][2]: 2  Choose operation:  1. Addition  2. Subtraction  3. Multiplication  4. Transpose (of A)  Enter your choice: 3  Resultant Matrix after Multiplication:  35 29 24  95 80 69  155 131 114 | Enter elements of matrix A (3x3):  A[0][0]: 9  A[0][1]: 8  A[0][2]: 7  A[1][0]: 6  A[1][1]: 5  A[1][2]: 4  A[2][0]: 3  A[2][1]: 2  A[2][2]: 1  Enter elements of matrix B (3x3):  B[0][0]: 1  B[0][1]: 1  B[0][2]: 1  B[1][0]: 1  B[1][1]: 1  B[1][2]: 1  B[2][0]: 1  B[2][1]: 1  B[2][2]: 1  Choose operation:  1. Addition  2. Subtraction  3. Multiplication  4. Transpose (of A)  Enter your choice: 1  Resultant Matrix after Addition:  10 9 8  7 6 5  4 3 2 |

Question 12:

Read and display Upper, Lower and Tri-diagonal matrices

Source Code:

|  |  |
| --- | --- |
| // Read and Display Upper, Lower and Tri-diagonal Triangle.  #include <stdio.h>  #include <stdlib.h> // Required for malloc()  // Function to input a dynamically allocated matrix  void inputMatrix(int SIZE, int \*\*matrix) {  printf("Enter elements of the matrix (%dx%d):\n", SIZE, SIZE);  for (int i = 0; i < SIZE; i++) {  for (int j = 0; j < SIZE; j++) {  printf("matrix[%d][%d]: ", i, j);  scanf("%d", &matrix[i][j]);  }  }  }  // Function to print a dynamically allocated matrix  void printMatrix(int SIZE, int \*\*matrix) {  printf("\nMatrix is:\n");  for (int i = 0; i < SIZE; i++) {  for (int j = 0; j < SIZE; j++) {  printf("%4d ", matrix[i][j]);  }  printf("\n");  }  }  // Function to print lower triangle matrix  void lowerTri(int SIZE, int \*\*matrix){  printf("\nLower Triangular Matrix:\n");  for (int i = 0; i < SIZE; i++) {  for (int j = 0; j < SIZE; j++) {  if (i>=j){  printf("%4d ", matrix[i][j]);  }  }  printf("\n");  }  }  // Function to print upper triangle matrix  void upperTri(int SIZE, int \*\*matrix){  printf("\nUpper Triangular Matrix:\n");  for (int i = 0; i < SIZE; i++) {  for (int j = 0; j < SIZE; j++) {  if (i<=j){  printf("%4d ", matrix[i][j]);  }  else{  printf(" ");  }  }  printf("\n");  }  } | // Function to print tri diagonal matrix  void triDiagonal(int SIZE, int \*\*matrix){  printf("\nTri Diagonal Matrix:\n");  for (int i = 0; i < SIZE; i++) {  for (int j = 0; j < SIZE; j++) {  if (i==j || i-1 == j || i+1 == j){  printf("%4d ", matrix[i][j]);  }  else{  printf(" ");  }  }  printf("\n");  }  }  // Main Function  int main() {  int SIZE;    printf("Enter Dimension of Square Matrix: ");  scanf("%d", &SIZE);  // Dynamically allocate 2D array  int \*\*matrix = (int \*\*)malloc(SIZE \* sizeof(int \*));  for (int i = 0; i < SIZE; i++) {  matrix[i] = (int \*)malloc(SIZE \* sizeof(int));  }  inputMatrix(SIZE, matrix);  printMatrix(SIZE, matrix);  lowerTri(SIZE, matrix);  upperTri(SIZE, matrix);  triDiagonal(SIZE, matrix);  // Free allocated memory  for (int i = 0; i < SIZE; i++) {  free(matrix[i]);  }  free(matrix);  return 0;  } |

Output:

|  |
| --- |
| Enter Dimension of Square Matrix: 3  Enter elements of the matrix (3x3):  matrix[0][0]: 1  matrix[0][1]: 2  matrix[0][2]: 3  matrix[1][0]: 4  matrix[1][1]: 5  matrix[1][2]: 6  matrix[2][0]: 7  matrix[2][1]: 8  matrix[2][2]: 9  Matrix is:  1 2 3  4 5 6  7 8 9  Lower Triangular Matrix:  1  4 5  7 8 9  Upper Triangular Matrix:  1 2 3  5 6  9  Tri Diagonal Matrix:  1 2  4 5 6  8 9 |

Question 13:

Implement sparse matrices using 3-tuple notation.

Source Code:

|  |  |
| --- | --- |
| // Implementing Sparse Matrix using 3-tuple Representation  #include <stdio.h>  #include <stdlib.h>  // // Global Variables  // int \*\*sparse;  // int \*\*matrix;  // Function to input a dynamically allocated matrix  void inputMatrix(int rowSIZE, int colSIZE, int \*\*matrix) {  printf("Enter elements of the matrix (%dx%d):\n", rowSIZE, colSIZE);  for(int i = 0; i < rowSIZE; i++) {  for(int j = 0; j < colSIZE; j++) {  printf("matrix[%d][%d]: ", i, j);  scanf("%d", &matrix[i][j]);  }  }  }  // Function to print a dynamically allocated matrix  void printMatrix(int rowSIZE, int colSIZE, int \*\*matrix) {  printf("Matrix is:\n");  for(int i = 0; i < rowSIZE; i++) {  for(int j = 0; j < colSIZE; j++) {  printf("%4d ", matrix[i][j]);  }  printf("\n");  }  }  // Function to create a Sparse Matrix  void toSparseMatrix(int rowSIZE, int colSIZE, int \*\*matrix){  int nonZeros = 0;  for(int i=0; i<rowSIZE; i++){  for(int j=0; j<colSIZE; j++){  if(matrix[i][j] != 0){  nonZeros++;  }  }  }  // Allocating Memory for Sparse Matrix  int \*\*sparse = (int \*\*)malloc((nonZeros+1) \* sizeof(int \*));  for(int i=0; i<nonZeros+1; i++){  sparse[i] = (int \*)malloc(3 \* sizeof(int));  }  // Storing dimentions and no. of nonZeros in First row  sparse[0][0] = rowSIZE;  sparse[0][1] = colSIZE;  sparse[0][2] = nonZeros;  // Storing Values in Sparse Matrix  int k=1; // k is index 1 of sparse array as 0th index contains data of og matrix  for(int i=0; i<rowSIZE; i++){  for(int j=0; j<colSIZE; j++){  if(matrix[i][j] != 0){  sparse[k][0] = i;  sparse[k][1] = j;  sparse[k][2] = matrix[i][j];  k++;  }  }  }  printf("Sparsed ");  printMatrix(nonZeros+1, 3, sparse);    }  // Coverting sparse to normal matrix  void toNormalMatrix(int rowCount, int \*\*sparse){  int rowSIZE = sparse[0][0];  int colSIZE = sparse[0][1];  // Dynamically allocate 2D array  int \*\*matrix = (int \*\*)malloc(rowSIZE \* sizeof(int \*));  for(int i = 0; i < rowSIZE; i++) {  matrix[i] = (int \*)malloc(colSIZE \* sizeof(int));  } | // Initialising Matix to 0  for(int i=0; i<rowSIZE; i++){  for(int j=0; j<colSIZE; j++){  matrix[i][j] = 0;  }  }  // Fetching and Placing non-zero values  for(int i=1; i<rowCount+1; i++){  matrix[sparse[i][0]][sparse[i][1]] = sparse[i][2];  }  printMatrix(rowSIZE, colSIZE, matrix);      }  // Main Driver Function  int main() {  int rowSIZE, colSIZE;  int choice;  printf("\nNormal to Sparse Matrix (1)\n");  printf("Sparse to Normal Matrix (2)\n");  printf("Enter Choice: ");  scanf("%d", &choice);  if(choice == 1){  printf("Enter Number of Rows: ");  scanf("%d", &rowSIZE);  printf("Enter Number of Columns: ");  scanf("%d", &colSIZE);  // Dynamically allocate 2D array  int \*\*matrix = (int \*\*)malloc(rowSIZE \* sizeof(int \*));  for(int i = 0; i < rowSIZE; i++) {  matrix[i] = (int \*)malloc(colSIZE \* sizeof(int));  }  inputMatrix(rowSIZE, colSIZE, matrix);  printMatrix(rowSIZE, colSIZE, matrix);  toSparseMatrix(rowSIZE, colSIZE, matrix);  // Free allocated memory  for(int i = 0; i < rowSIZE; i++) {  free(matrix[i]);  }  free(matrix);  }  else if(choice == 2){  int nonZero;  printf("\nEnter Number of non-zero values: ");  scanf("%d", &nonZero);  // Dynamically allocate 2D array  int \*\*sparse = (int \*\*)malloc((nonZero+1) \* sizeof(int \*));  for(int i = 0; i < nonZero+1; i++) {  sparse[i] = (int \*)malloc(3 \* sizeof(int));  }  inputMatrix(nonZero+1, 3, sparse);  toNormalMatrix(nonZero, sparse);  for(int i = 0; i < nonZero+1; i++) {  free(sparse[i]);  }  free(sparse);  }  else{  printf("\nINVALID INPUT\n");  }    return 0;  } |

Output:

|  |  |
| --- | --- |
| Normal to Sparse Matrix (1)  Sparse to Normal Matrix (2)  Enter Choice: 1  Enter Number of Rows: 4  Enter Number of Columns: 3  Enter elements of the matrix (4x3):  matrix[0][0]: 0  matrix[0][1]: 0  matrix[0][2]: 0  matrix[1][0]: 0  matrix[1][1]: 0  matrix[1][2]: 5  matrix[2][0]: 0  matrix[2][1]: 0  matrix[2][2]: 8  matrix[3][0]: 0  matrix[3][1]: 0  matrix[3][2]: 3  Matrix is:  0 0 0  0 0 5  0 0 8  0 0 3  Sparsed Matrix is:  4 3 3  1 2 5  2 2 8  3 2 3 | Normal to Sparse Matrix (1)  Sparse to Normal Matrix (2)  Enter Choice: 2  Enter Number of non-zero values: 3  Enter elements of the Sparse Matrix (4x3):  matrix[0][0]: 4  matrix[0][1]: 3  matrix[0][2]: 3  matrix[1][0]: 1  matrix[1][1]: 2  matrix[1][2]: 5  matrix[2][0]: 2  matrix[2][1]: 2  matrix[2][2]: 8  matrix[3][0]: 3  matrix[3][1]: 2  matrix[3][2]: 3  Matrix is:  0 0 0  0 0 5  0 0 8  0 0 3 |

Question 14:

Write a Program to implement Selection Sort.

Source Code:

|  |  |
| --- | --- |
| #include <stdio.h>  #include <stdlib.h> // For malloc and free  void selectionSort(int arr[], int n) {  int i, j, minIndex, temp;    for (i = 0; i < n - 1; i++) {  minIndex = i;    // Find the minimum element in the unsorted part of the array  for (j = i + 1; j < n; j++) {  if (arr[j] < arr[minIndex]) {  minIndex = j;  }  }    // Swap the found minimum element with the element at i  if (minIndex != i) {  temp = arr[i];  arr[i] = arr[minIndex];  arr[minIndex] = temp;  }  }  }  void printArray(int arr[], int n) {  for (int i = 0; i < n; i++) {  printf("%d ", arr[i]);  }  printf("\n");  } | int main() {  int n;    // Prompt the user to enter the size of the array  printf("Enter the number of elements: ");  scanf("%d", &n);    // Dynamically allocate memory for the array based on the user input  int \*arr = (int \*)malloc(n \* sizeof(int)); // Using malloc for dynamic memory allocation    if (arr == NULL) { // Check if memory allocation was successful  printf("Memory allocation failed!\n");  return 1;  }    // Prompt the user to input the elements of the array  printf("Enter %d elements: ", n);  for (int i = 0; i < n; i++) {  scanf("%d", &arr[i]);  }    printf("Original Array: ");  printArray(arr, n);    selectionSort(arr, n);    printf("Sorted Array: ");  printArray(arr, n);    // Free the dynamically allocated memory  free(arr);    return 0;  } |

Output:

|  |
| --- |
| Enter the number of elements: 5  Enter 5 elements: 4  6  8  3  9  Original Array: 4 6 8 3 9  Sorted Array: 3 4 6 8 9 |