**PS2-Week 1- 2 Qsn 1**

#include <stdio.h>

// Windows: Shift + Alt + F - for formatting

void populateArray(int arr[], int arrSize);

void printArray(int arr[], int arrSize);

void selectionSortArray(int arr[], int arrSize);

int get2ndLargest(int arr[], int arrSize);

void swap(int \*arr, int posn1, int posn2);

int main()

{

int arrSize, refPosn;

printf("\nEnter number of integers you will input : ");

scanf("%d", &arrSize);

int arr[arrSize];

populateArray(arr, arrSize);

printArray(arr, arrSize);

int secondHighest = get2ndLargest(arr, arrSize);

printf("\nSecond Highest : %d", arr[1]);

printf("\n");

return 0;

}

void populateArray(int arr[], int arrSize)

{

printf("\nEnter array elements : \n");

for (int i = 0; i < arrSize; i++)

{

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

}

}

void printArray(int arr[], int arrSize)

{

printf("\nThe array elements are : ");

for (int i = 0; i < arrSize; i++)

{

printf("%4d", arr[i]);

}

}

int get2ndLargest(int arr[], int arrSize)

{

int secondHighest;

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

{

for (int j = i + 1; j < arrSize; j++)

{

if (arr[i] < arr[j])

swap(arr, i, j);

}

}

secondHighest = arr[1];

return secondHighest;

}

void swap(int arr[], int posn1, int posn2)

{

int temp = arr[posn1];

arr[posn1] = arr[posn2];

arr[posn2] = temp;

}

void selectionSortArray(int arr[], int arrSize)

{

int refPosn;

for (int i = 0; i < arrSize - 1; i++)

{

refPosn = i;

for (int j = i + 1; j < arrSize; j++)

{

// if (arr[refPosn] < arr[j]) // Descending order

if (arr[refPosn] > arr[j]) // Ascending order

refPosn = j;

}

if (refPosn != i)

{

swap(arr, refPosn, i);

}

}

}

**PS2-Week 1- 2 Qsn 2**

#include <stdio.h>

#include <stdbool.h>

// Windows: Shift + Alt + F - for formatting

void populateArray(int arr[], int arrSize);

void printArray(int arr[], int arrSize);

// void selectionSortArray(int arr[], int arrSize);

void getSortedArray(int arr[], int arrSize, bool asc);

void swap(int \*arr, int posn1, int posn2);

int main()

{

int arrSize, refPosn;

printf("\nEnter number of integers you will input : ");

scanf("%d", &arrSize);

int arr[arrSize];

populateArray(arr, arrSize);

printArray(arr, arrSize);

bool asc = true;

getSortedArray(arr, arrSize, asc);

printf("\nArray in Ascending order : ");

printArray(arr, arrSize);

asc = false;

printf("\nArray inDescending order : ");

getSortedArray(arr, arrSize, asc);

printArray(arr, arrSize);

printf("\n");

return 0;

}

void populateArray(int arr[], int arrSize)

{

printf("\nEnter array elements : \n");

for (int i = 0; i < arrSize; i++)

{

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

}

}

void printArray(int arr[], int arrSize)

{

printf("\nThe array elements are : ");

for (int i = 0; i < arrSize; i++)

{

printf("%4d", arr[i]);

}

}

void getSortedArray(int arr[], int arrSize, bool asc)

{

int secondHighest;

for (int i = 0; i < arrSize - 1; i++)

{

for (int j = i + 1; j < arrSize; j++)

{

if (asc)

{

if (arr[i] > arr[j]) // Ascending order

swap(arr, i, j);

}

else

{

if (arr[i] < arr[j]) // Descending order

swap(arr, i, j);

}

}

}

}

void swap(int arr[], int posn1, int posn2)

{

int temp = arr[posn1];

arr[posn1] = arr[posn2];

arr[posn2] = temp;

}

/\*

void selectionSortArray(int arr[], int arrSize)

{

int refPosn;

for (int i = 0; i < arrSize - 1; i++)

{

refPosn = i;

for (int j = i + 1; j < arrSize; j++)

{

// if (arr[refPosn] < arr[j]) // Descending order

if (arr[refPosn] > arr[j]) // Ascending order

refPosn = j;

}

if (refPosn != i)

{

swap(arr, refPosn, i);

}

}

}

\*/

**PS2-Week 1- 2 Qsn 3**

#include <stdio.h>

#include <stdbool.h>

// Windows: Shift + Alt + F - for formatting

void populateArray(int arr[], int arrSize);

void printArray(int arr[], int arrSize);

void getIndexOfValueToSearch(int arr[], int aSz);

int main()

{

int arrSize, refPosn;

printf("\nEnter number of integers you will input : ");

scanf("%d", &arrSize);

int arr[arrSize];

populateArray(arr, arrSize);

printArray(arr, arrSize);

getIndexOfValueToSearch(arr, arrSize);

printf("\n");

return 0;

}

void getIndexOfValueToSearch(int arr[], int aSz)

{

int theVal;

bool found = false;

printf("\nEnter the element value in the Array to search for its position : ");

scanf("%d", &theVal);

for (int i = 0; i < aSz; i++)

{

if (theVal == arr[i])

{

printf("\nThe element : %d exists at index position %d : ", theVal, i);

return;

}

else if (!found && (i == (aSz - 1)))

{

printf("\nThe element value : %d entered does not exist in the array. Quitting!", theVal);

}

}

}

void populateArray(int arr[], int arrSize)

{

printf("\nEnter array elements : \n");

for (int i = 0; i < arrSize; i++)

{

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

}

}

void printArray(int arr[], int arrSize)

{

printf("\nThe array elements are : ");

for (int i = 0; i < arrSize; i++)

{

printf("%4d", arr[i]);

}

}**PS2-Week 1- 2 Qsn 4**

#include <stdio.h>

#include <stdbool.h>

// Windows: Shift + Alt + F - for formatting

void populateArray(int arr[], int arrSize);

void printArray(int arr[], int arrSize);

void processArrElements(int arr[], int aSz);

void printStats(int pos, int neg, int odd, int even, int zero);

int main()

{

int arrSize, refPosn;

printf("\nEnter number of integers you will input : ");

scanf("%d", &arrSize);

int arr[arrSize];

populateArray(arr, arrSize);

printArray(arr, arrSize);

processArrElements(arr, arrSize);

printf("\n");

return 0;

}

void processArrElements(int arr[], int aSz)

{

int pos = 0, neg = 0, odd = 0, even = 0, zero = 0;

for (int i = 0; i < aSz; i++)

{

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

{

zero += 1;

}

else if (arr[i] < 0)

{

neg += 1;

}

else if (arr[i] > 0)

{

pos += 1;

}

int rem = (arr[i] % 2);

if (0 == rem)

{

even += 1;

}

else

{

odd += 1;

}

}

even = even - zero;

printStats(pos, neg, odd, even, zero);

}

void printStats(int pos, int neg, int odd, int even, int zero)

{

printf("\nNo of Zeros : %d", zero);

printf("\nNo of Positive entries : %d", pos);

printf("\nNo of Negative entries : %d", neg);

printf("\nNo of Odd entries : %d", odd);

printf("\nNo of Even entries : %d", even);

}

void populateArray(int arr[], int arrSize)

{

printf("\nEnter array elements : \n");

for (int i = 0; i < arrSize; i++)

{

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

}

}

void printArray(int arr[], int arrSize)

{

printf("\nThe array elements are : ");

for (int i = 0; i < arrSize; i++)

{

printf("%4d", arr[i]);

}

}

**PS2-Week 1- 2 Qsn 5**

#include <stdio.h>

#include <stdbool.h>

// Windows: Shift + Alt + F - for formatting

void populateArray(int arr[], int arrSize);

void printArray(int arr[], int arrSize);

void shiftArrElements(int arr[], int arrSz, int aTempArr[], int keySz);

int main()

{

int arrSize, key;

printf("\nEnter number of integers you will input : ");

scanf("%d", &arrSize);

int arr[arrSize];

populateArray(arr, arrSize);

printArray(arr, arrSize);

printf("\nEnter 'Key' value, the number of left shift of array values : ");

scanf("%d", &key);

key = key % arrSize;

int arrTemp[key];

shiftArrElements(arr, arrSize, arrTemp, key);

printArray(arr, arrSize);

printf("\n");

return 0;

}

void shiftArrElements(int arr[], int arrSz, int aTempArr[], int keySz)

{

int theArrSize = arrSz, theTempArrSize = keySz;

if (0 == theTempArrSize)

return;

for (int i = 0; i < theTempArrSize; i++)

{

aTempArr[i] = arr[i];

}

int theStartIndx = theTempArrSize;

int j = 0;

for (int i = theStartIndx; i < theArrSize; i++)

{

arr[j] = arr[i];

j++;

}

theStartIndx = theArrSize - theTempArrSize;

j = 0;

for (int i = theStartIndx; i < theArrSize; i++)

{

arr[i] = aTempArr[j];

j++;

}

}

void populateArray(int arr[], int arrSize)

{

printf("\nEnter array elements : \n");

for (int i = 0; i < arrSize; i++)

{

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

}

}

void printArray(int arr[], int arrSize)

{

printf("\nThe array elements are : ");

for (int i = 0; i < arrSize; i++)

{

printf("%4d", arr[i]);

}

}

**PS2-Week 1- 2 Qsn 6**

#include <stdio.h>

#include <stdbool.h>

#include <stdlib.h>

#include <time.h>

// Windows: Shift + Alt + F - for formatting

#define ARRSIZE 5

#define MAXNUM 10

void populate2DArray(int arrSize, int (\*arr)[arrSize]);

void populate2DUpperTriangularMxArray(int arrSize, int (\*arr)[arrSize], bool upper);

void printSumsOfTriangles(int arrSize, int (\*arr)[arrSize], bool upper);

void printArray(int arrSize, int (\*arr)[arrSize]);

void printSumOfDiagnolElements(int arrSize, int (\*arr)[arrSize]);

int main()

{

int arrSize;

time\_t theSeed;

/\* define the random number generator \*/

srand((unsigned)time(&theSeed)); // pass the srand() parameter

arrSize = ARRSIZE;

int arr[arrSize][arrSize];

populate2DArray(arrSize, arr);

printf("\nThe array elements are : \n");

printArray(arrSize, arr);

printSumOfDiagnolElements(arrSize, arr);

printf("\n");

bool upper = true; // false;

populate2DUpperTriangularMxArray(arrSize, arr, upper);

printf("\nThe array elements of Upper Triangular Matrix are : \n");

printArray(arrSize, arr);

printSumsOfTriangles(arrSize, arr, upper);

printf("\n");

upper = false;

populate2DUpperTriangularMxArray(arrSize, arr, upper);

printf("\nThe array elements of Lower Triangular Matrix are : \n");

printArray(arrSize, arr);

printSumsOfTriangles(arrSize, arr, upper);

printf("\n\n");

return 0;

}

void printSumsOfTriangles(int arrSize, int (\*arr)[arrSize], bool upper)

{

int nRows = arrSize;

int mCols = arrSize;

int sum = 0;

int startJ = 0, endJ = 0;

for (int i = 0; i < nRows; i++)

{

if (upper)

{

startJ = i;

endJ = mCols - 1;

}

else

{

startJ = 0;

endJ = i;

}

for (int j = startJ; j <= endJ; j++)

{

sum += \*(\*(arr + i) + j);

}

}

printf("\nThe Sum of elements of %s Triangular Matrix is : %d", (upper ? "Upper" : "Lower"), sum);

}

void populate2DUpperTriangularMxArray(int arrSize, int (\*arr)[arrSize], bool upper)

{

int nRows = arrSize;

int mCols = arrSize;

for (int i = 0; i < nRows; i++)

{

for (int j = 0; j < mCols; j++)

{

if (upper ? j < i : j > i)

{

\*(\*(arr + i) + j) = 0;

}

else

{

int aVal;

aVal = rand() % MAXNUM;

aVal++;

\*(\*(arr + i) + j) = aVal; // rand() % MAXNUM;

}

}

}

}

void printSumOfDiagnolElements(int arrSize, int (\*arr)[arrSize])

{

int nRows = arrSize;

int mCols = arrSize;

int sum = 0;

printf("\nThe Major Diagnol elements are : ");

for (int i = 0, j = 0; i < nRows; i++, j++)

{

sum += \*(\*(arr + i) + j);

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

}

printf("\nThe Sum of Major Diagnol elements is : %d", sum);

sum = 0;

printf("\n\nThe Minor Diagnol elements are : ");

for (int i = nRows - 1, j = 0; i >= 0; i--, j++)

{

sum += \*(\*(arr + i) + j);

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

}

printf("\nThe Sum of Minor Diagnol elements is : %d", sum);

}

void populate2DArray(int arrSize, int (\*arr)[arrSize])

{

int nRows = arrSize;

int mCols = arrSize;

for (int i = 0; i < nRows; i++)

{

for (int j = 0; j < mCols; j++)

{

\*(\*(arr + i) + j) = rand() % MAXNUM;

}

}

}

void printArray(int arrSize, int (\*arr)[arrSize])

{

int nRows = arrSize;

int mCols = arrSize;

for (int i = 0; i < nRows; i++)

{

for (int j = 0; j < mCols; j++)

{

// printf("%4d", \*( \*(arr + i) + j) );

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

}

printf("\n\n");

}

}

**PS2-Week 1- 2 Qsn 7**

#include <stdio.h>

#include <stdbool.h>

#include <stdlib.h>

#include <time.h>

// Windows: Shift + Alt + F - for formatting

#define ARRSIZE 5

#define MAXNUM 5

#define MAXROWS 6

#define MAXCOLS 5

void populate2DArray(int rows, int cols, int (\*arr)[cols]);

void checkForZeros(int rows, int cols, int (\*arr)[cols];);

void makeItZeros(int rows, int cols, int (\*arr)[cols], int aRow, int aCol);

void printArray(int rows, int cols, int (\*arr)[cols]);

int main()

{

int nRows, mCols;

printf("\nEnter number of Rows in the Matrix : ");

scanf("%d", &nRows);

printf("\nEnter number of Columns in the Matrix : ");

scanf("%d", &mCols);

int arr[nRows][mCols];

populate2DArray(nRows, mCols, arr);

printf("\nThe Array elements entered are : \n");

printf("\n");

printArray(nRows, mCols, arr);

printf("\n");

checkForZeros(nRows, mCols, arr);

printf("\nThe Array elements entered are : \n");

printArray(nRows, mCols, arr);

printf("\n\n");

return 0;

}

void populate2DArray(int rows, int cols, int (\*arr)[cols])

{

int nRows = rows;

int mCols = cols;

printf("\nEnter array elements row wise : \n");

for (int i = 0; i < nRows; i++)

{

for (int j = 0; j < mCols; j++)

{

scanf("%d", (\*(arr + i) + j));

}

}

}

void checkForZeros(int rows, int cols, int (\*arr)[cols])

{

int nRows = rows;

int mCols = cols;

int aVal, theRowNo, theColNo;

int zeroCnt = 0;

for (int i = 0; i < nRows; i++)

{

for (int j = 0; j < mCols; j++)

{

if (0 == \*(\*(arr + i) + j))

{

theRowNo = i;

theColNo = j;

zeroCnt += 1;

}

}

}

if (0 == zeroCnt)

{

printf("\nNo 0 Value element found in Matrix! \n");

}

else if (zeroCnt > 1)

{

printf("\nMore than one 0 Value element found in Matrix! \n");

}

else

{

printf("\nFound a single 0 value element at M(x , y) : M(%d , %d) \n", theRowNo + 1, theColNo + 1);

makeItZeros(nRows, mCols, arr, theRowNo, theColNo);

}

}

void makeItZeros(int rows, int cols, int (\*arr)[cols], int aRow, int aCol)

{

int nRows = rows;

int mCols = cols;

int theRowNo = aRow, theColNo = aCol;

int i = theRowNo;

printf("\nModified to 0 the entire Row No :%d \n", theRowNo + 1);

for (int j = 0; j < mCols; j++)

{

\*(\*(arr + i) + j) = 0;

}

int j = theColNo;

printf("\nModified to 0 the entire Col No :%d \n", theColNo + 1);

for (int i = 0; i < nRows; i++)

{

\*(\*(arr + i) + j) = 0;

}

}

void printArray(int rows, int cols, int (\*arr)[cols])

{

int nRows = rows;

int mCols = cols;

printf("\n");

for (int i = 0; i < nRows; i++)

{

for (int j = 0; j < mCols; j++)

{

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

}

printf("\n\n");

}

}

**PS2-Week 1- 2 Qsn 8**

#include <stdio.h>

#include <stdlib.h>

#include <stdbool.h>

#include <time.h>

#define MAXNUM 25

void populateArray(int \*arr, int aSz);

void displayArray(int \*arr, int aSz);

void sortTheArray(int \*arr, int aSz);

void mergeSortedArrays(int \*arr1, int \*aSz1, int \*arr2, int aSz2);

void swap(int \*arr, int posn1, int posn2);

int main()

{

time\_t theSeed;

srand((unsigned)time(&theSeed));

int arrSize1, arrSize2;

printf("\nEnter No of elements in 1st array : ");

scanf("%d", &arrSize1);

printf("\nEnter No of elements in 2nd array : ");

scanf("%d", &arrSize2);

int arr1[arrSize1];

int arr2[arrSize2];

populateArray(arr1, arrSize1);

printf("\n");

printf("\nThe Array 1 elements are : ");

displayArray(arr1, arrSize1);

sortTheArray(arr1, arrSize1);

printf("\nSorted Array 1 elements are : ");

displayArray(arr1, arrSize1);

populateArray(arr2, arrSize2);

printf("\n");

printf("\nThe Array 2 elements are : ");

displayArray(arr2, arrSize2);

sortTheArray(arr2, arrSize2);

printf("\nSorted Array 2 elements are : ");

displayArray(arr2, arrSize2);

mergeSortedArrays(arr1, &arrSize1, arr2, arrSize2);

printf("\n");

printf("\nMerged Sorted Array elements are : ");

displayArray(arr1, arrSize1);

}

void mergeSortedArrays(int \*arr1, int \*aSz1, int \*arr2, int aSz2)

{

int theSz1 = \*aSz1;

int theSz2 = aSz2;

int newArr[theSz1 + theSz2], newIndx = 0;

int arr1Indx = 0, arr2Indx = 0;

for (arr1Indx = 0, arr2Indx = 0; arr1Indx < theSz1 && arr2Indx < theSz2;)

{

if (\*(arr1 + arr1Indx) <= \*(arr2 + arr2Indx))

{

\*(newArr + newIndx) = \*(arr1 + arr1Indx);

arr1Indx++;

}

else

{

\*(newArr + newIndx) = \*(arr2 + arr2Indx);

arr2Indx++;

}

newIndx++;

}

for (int i = arr1Indx; i < theSz1; i++)

{

\*(newArr + newIndx) = \*(arr1 + i);

newIndx++;

}

for (int j = arr2Indx; j < theSz2; j++)

{

\*(newArr + newIndx) = \*(arr2 + j);

newIndx++;

}

printf("\n");

for (int i = 0; i < newIndx; i++)

{

\*(arr1 + i) = \*(newArr + i);

}

\*aSz1 = newIndx;

}

void populateArray(int \*arr, int aSz)

{

int theSize = aSz;

int theVal = 0;

for (int i = 0; i < theSize; i++)

{

theVal = rand() % MAXNUM;

theVal++;

\*(arr + i) = theVal;

}

}

void displayArray(int \*arr, int aSz)

{

int theSize = aSz;

for (int i = 0; i < theSize; i++)

{

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

}

}

void sortTheArray(int \*arr, int aSz)

{

int theSize = aSz;

int refPosn = 0;

for (int i = 0; i < theSize - 1; i++)

{

refPosn = i;

for (int j = i + 1; j < theSize; j++)

{

if (\*(arr + refPosn) > \*(arr + j))

{

refPosn = j;

}

}

if (refPosn != i)

{

swap(arr, i, refPosn);

}

}

}

void swap(int \*arr, int posn1, int posn2)

{

int thePosn1, thePosn2, temp;

temp = \*(arr + posn1);

\*(arr + posn1) = \*(arr + posn2);

\*(arr + posn2) = temp;

}

**PS2-Week 1- 2 Qsn 9**

#include <stdio.h>

#include <stdlib.h>

#include <stdbool.h>

#include <time.h>

#define MAXNUM 25

void populateArray(int \*arr, int aSz);

void displayArray(int \*arr, int aSz);

void removeDuplicatesInArray(int \*arr, int \*aSz);

void shiftLeftArrElements(int \*arr, int arrSz, int aStartPosn);

int main()

{

time\_t theSeed;

srand((unsigned)time(&theSeed));

int arrSize;

printf("\nEnter No of elements in array : ");

scanf("%d", &arrSize);

int arr[arrSize];

populateArray(arr, arrSize);

printf("\n");

printf("\nThe Array elements are : ");

displayArray(arr, arrSize);

removeDuplicatesInArray(arr, &arrSize);

printf("\n");

printf("\nThe Array elements after removal of duplicates : ");

displayArray(arr, arrSize);

removeDuplicatesInArray(arr, &arrSize);

printf("\n");

}

void populateArray(int \*arr, int aSz)

{

int theSize = aSz;

int theVal = 0;

for (int i = 0; i < theSize; i++)

{

do

{

theVal = rand() % MAXNUM; /\* code \*/

} while (0 == theVal);

\*(arr + i) = theVal;

}

}

void displayArray(int \*arr, int aSz)

{

int theSize = aSz;

for (int i = 0; i < theSize; i++)

{

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

}

}

void removeDuplicatesInArray(int \*arr, int \*aSz)

{

int theSize = \*aSz;

int refPosn = 0;

bool found = false;

for (int i = 0; i < theSize - 1; i++)

{

for (int j = i + 1; j < theSize; j++)

{

if (\*(arr + i) == \*(arr + j))

{

found = true;

shiftLeftArrElements(arr, theSize, j);

theSize--;

j--;

}

}

}

\*aSz = theSize;

if (!found)

printf("\nNo duplicates exists in the Array\n");

}

void shiftLeftArrElements(int \*arr, int arrSz, int aStartPosn)

{

int theArrSize = arrSz;

int theStartIndx = aStartPosn;

int j = theStartIndx;

for (int i = j + 1; i < theArrSize; i++)

{

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

j++;

}

}

**PS2-Week 1- 2 Qsn 10**

#include <stdio.h>

#include <stdlib.h>

#include <stdbool.h>

#include <time.h>

#define MAXNUM 25

// void populateArray(int \*arr, int arrSize);

void populateArray2(int \*arr, int aSz);

void displayArray(int \*arr, int aSz);

void countPosNegVal(int \*arr, int arrSize, int \*pos, int \*neg, int \*zero);

void arrangeAlternateSign(int \*arr, int \*arrSize);

void segregateArrays(int \*arr, int arrSize, int \*posArr, int \*negArr, int \*zeroArr);

void mergeArrays(int \*posArr, int \*pos, int \*negArr, int neg, int \*zeroArr, int zero);

int main()

{

time\_t theSeed;

srand((unsigned)time(&theSeed));

int arrSize;

printf("\nEnter No of elements in array : ");

scanf("%d", &arrSize);

int arr[arrSize];

populateArray2(arr, arrSize);

printf("\n");

printf("\nThe Array elements are : ");

displayArray(arr, arrSize);

printf("\n");

arrangeAlternateSign(arr, &arrSize);

printf("\n");

printf("\nThe Array elements alternatively arranged : ");

displayArray(arr, arrSize);

printf("\n");

}

void arrangeAlternateSign(int \*arr, int \*arrSize)

{

int theSize = \*arrSize;

int pos = 0, neg = 0, zero = 0;

countPosNegVal(arr, \*arrSize, &pos, &neg, &zero);

int posArr[pos], negArr[neg], zeroArr[zero];

segregateArrays(arr, \*arrSize, posArr, negArr, zeroArr);

printf("\n");

printf("\n");

printf("\n");

mergeArrays(posArr, &pos, negArr, neg, zeroArr, zero);

for (int i = 0; i < theSize; i++)

{

\*(arr + i) = \*(posArr + i);

}

\*arrSize = pos;

}

void segregateArrays(int \*arr, int arrSize, int \*posArr, int \*negArr, int \*zeroArr)

{

int p = 0, n = 0, z = 0;

for (int i = 0; i < arrSize; i++)

{

if (\*(arr + i) == 0)

{

\*(zeroArr + z) = \*(arr + i);

z++;

}

else if (\*(arr + i) > 0)

{

\*(posArr + p) = \*(arr + i);

p++;

}

else

{

\*(negArr + n) = \*(arr + i);

n++;

}

}

}

void countPosNegVal(int \*arr, int arrSize, int \*pos, int \*neg, int \*zero)

{

for (int i = 0; i < arrSize; i++)

{

if (\*(arr + i) == 0)

{

(\*zero)++;

}

else if (\*(arr + i) > 0)

{

(\*pos)++;

}

else

{

(\*neg)++;

}

}

}

void mergeArrays(int \*posArr, int \*pos, int \*negArr, int neg, int \*zeroArr, int zero)

{

int theSz1 = \*pos;

int theSz2 = neg;

int theSz3 = zero;

int newArr[theSz1 + theSz2 + theSz3], newIndx = 0;

int posIndx = 0, negIndx = 0, zeroIndx = 0;

for (posIndx = 0, negIndx = 0; posIndx < theSz1 && negIndx < theSz2;)

{

\*(newArr + newIndx) = \*(posArr + posIndx);

\*(newArr + newIndx + 1) = \*(negArr + negIndx);

posIndx++;

negIndx++;

newIndx += 2;

}

for (int i = posIndx; i < theSz1; i++)

{

\*(newArr + newIndx) = \*(posArr + i);

newIndx++;

}

for (int j = negIndx; j < theSz2; j++)

{

\*(newArr + newIndx) = \*(negArr + j);

newIndx++;

}

for (int j = zeroIndx; j < theSz3; j++)

{

\*(newArr + newIndx) = \*(zeroArr + j);

newIndx++;

}

for (int i = 0; i < newIndx; i++)

{

\*(posArr + i) = \*(newArr + i);

}

\*pos = newIndx;

}

/\*

void populateArray(int \*arr, int arrSize)

{

printf("\nEnter array elements : \n");

for (int i = 0; i < arrSize; i++)

{

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

}

}

\*/

void populateArray2(int \*arr, int aSz)

{

int theSize = aSz;

int theVal = 0;

for (int i = 0; i < theSize; i++)

{

theVal = rand() % MAXNUM;

theVal++;

int pos = rand() % 2;

if (0 == pos)

{

theVal \*= -1;

}

\*(arr + i) = theVal;

}

}

void displayArray(int \*arr, int aSz)

{

int theSize = aSz;

for (int i = 0; i < theSize; i++)

{

printf("%4d\t", \*(arr + i));

}

}

**PS2-Week 1- 2 Qsn 11**

#include <stdio.h>

// Windows: Shift + Alt + F - for formatting

void populateArray(int arr[], int arrSize);

void printArray(int arr[], int arrSize);

void swap(int \*arr, int posn1, int posn2);

void sortTheArray(int \*arr, int aSz);

int sumArrayElements(int arr[], int aSz, int arrSumEl[], int \*aSzNew);

int main()

{

int arrSize;

printf("\nEnter number of integers you will input : ");

scanf("%d", &arrSize);

int arr[arrSize];

int arrSumEl[arrSize];

populateArray(arr, arrSize);

printf("\nThe array elements are : ");

printArray(arr, arrSize);

sortTheArray(arr, arrSize);

printf("\n");

printf("\nThe Sorted array elements are : ");

printArray(arr, arrSize);

printf("\n");

int theSz;

int sum = sumArrayElements(arr, arrSize, arrSumEl, &theSz);

printf("\n\nThe lowest sum found = %d", sum);

printf("\nformed by the Subset (");

printArray(arrSumEl, theSz);

printf(")");

if (sum < 2)

{

printf("\n\nNo subset(containing minimum 2 elements) found whose sum is atleast 2\n\n");

}

else

{

printf("\n\nThe smallest positive integer that cannot be represented as");

printf("\nthe sum of any subset(containing minimum 2 elements) of the given array is : %d\n\n", (sum - 1));

}

printf("\n");

return 0;

}

int sumArrayElements(int arr[], int aSz, int arrSumEl[], int \*aSzNew)

{

int theSz = aSz;

int sum = 0;

int i = 0;

int j = 0;

\*aSzNew = 0;

j = i + 1;

sum = \*(arr + i) + \*(arr + j);

arrSumEl[(\*aSzNew)++] = \*(arr + i);

arrSumEl[(\*aSzNew)++] = \*(arr + j);

if (sum >= 2)

{

return sum;

}

else

{

j++;

do

{

sum += \*(arr + j);

arrSumEl[(\*aSzNew)++] = \*(arr + j);

j++;

} while (sum < 2 && j < theSz);

}

return sum;

}

void populateArray(int arr[], int arrSize)

{

printf("\nEnter array elements : \n");

for (int i = 0; i < arrSize; i++)

{

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

}

}

void printArray(int arr[], int arrSize)

{

for (int i = 0; i < arrSize; i++)

{

printf("%4d%s", arr[i], (i < (arrSize - 1) ? "," : ""));

}

}

void sortTheArray(int \*arr, int aSz)

{

int theSize = aSz;

int refPosn = 0;

for (int i = 0; i < theSize - 1; i++)

{

refPosn = i;

for (int j = i + 1; j < theSize; j++)

{

if (\*(arr + refPosn) > \*(arr + j)) // Descending sort

// if (\*(arr + refPosn) < \*(arr + j))//Ascending sort

{

refPosn = j;

}

}

if (refPosn != i)

{

swap(arr, i, refPosn);

}

}

}

void swap(int arr[], int posn1, int posn2)

{

int temp = arr[posn1];

arr[posn1] = arr[posn2];

arr[posn2] = temp;

}

**PS2-Week 1- 2 Qsn 12**

#include <stdio.h>

#include <stdlib.h>

#include <stdbool.h>

#include <time.h>

#define MAXNUM 25

void populateArray(int \*arr, int aSz);

void displayArray(int \*arr, int aSz);

void sortTheArray(int \*arr, int aSz);

void swap(int \*arr, int posn1, int posn2);

bool elementFoundInArray(int arr[], int aSz, int aVal);

int findMinOf3(int a1, int a2, int a3, int \*arrSz4);

void checkEachArr(int arr1[], int aSz1, int arr2[], int aSz2, int arr3[], int aSz3, int arr4[], int \*aSz4);

int main()

{

time\_t theSeed;

srand((unsigned)time(&theSeed));

int arrSize1, arrSize2, arrSize3, arrSize4;

printf("\nEnter No of elements in 1st array : ");

scanf("%d", &arrSize1);

printf("\nEnter No of elements in 2nd array : ");

scanf("%d", &arrSize2);

printf("\nEnter No of elements in 3rd array : ");

scanf("%d", &arrSize3);

printf("\nGenerating the three arrays with random numbers.... ");

int arr1[arrSize1];

int arr2[arrSize2];

int arr3[arrSize3];

// int arr1[5] = {2,4,5,6, 9};

// int arr2[7] = {5,3,2,8, 3, 7, 0};

// int arr3[9] = {5,1, 6, 3, 2, 8, 9, 0, 4};

populateArray(arr1, arrSize1);

printf("\n");

sortTheArray(arr1, arrSize1);

printf("\nSorted Array 1 elements are : ");

displayArray(arr1, arrSize1);

populateArray(arr2, arrSize2);

printf("\n");

sortTheArray(arr2, arrSize2);

printf("\nSorted Array 2 elements are : ");

displayArray(arr2, arrSize2);

printf("\n");

populateArray(arr3, arrSize3);

printf("\n");

sortTheArray(arr3, arrSize3);

printf("\nSorted Array 3 elements are : ");

displayArray(arr3, arrSize3);

printf("\n");

int minSize = findMinOf3(arrSize1, arrSize2, arrSize3, &arrSize4);

int arr4[arrSize4];

switch (minSize)

{

case 1:

checkEachArr(arr1, arrSize1, arr2, arrSize2, arr3, arrSize3, arr4, &arrSize4);

break;

case 2:

checkEachArr(arr2, arrSize2, arr1, arrSize1, arr3, arrSize3, arr4, &arrSize4);

break;

case 3:

checkEachArr(arr3, arrSize3, arr2, arrSize2, arr1, arrSize1, arr4, &arrSize4);

break;

default:

break;

}

printf("\n");

if (arrSize4)

{

printf("\nThe Common elements in all 3 array are : ");

displayArray(arr4, arrSize4);

}

printf("\n");

return 0;

}

void checkEachArr(int arr1[], int aSz1, int arr2[], int aSz2, int arr3[], int aSz3, int arr4[], int \*aSz4)

{

int theSize1 = aSz1;

bool found = false;

int comCnt = 0;

int indx4 = 0;

for (int i = 0; i < theSize1; i++)

{

int theCurEl = arr1[i];

if (elementFoundInArray(arr2, aSz2, theCurEl) && elementFoundInArray(arr3, aSz3, theCurEl))

{

found = true;

comCnt++;

if (!elementFoundInArray(arr4, \*aSz4, theCurEl))

{

arr4[indx4] = theCurEl;

indx4++;

\*aSz4 = indx4;

}

}

}

if (!found)

printf("\nZero Common elements: \n");

if (comCnt)

printf("\nNo of Common elements are : %d\n", comCnt);

}

int findMinOf3(int a1, int a2, int a3, int \*arrSz4)

{

if (a1 <= a2) // a1 <= a2 => a1 and a3

{

if (a1 <= a3) // a1 < a3

{

\*arrSz4 = a1;

return 1;

}

else

{

\*arrSz4 = a3;

return 3; // else a3 < a1

}

}

else // a2 < a1 => a2 and a3

{

if (a2 <= a3) // a2 < a3

{

\*arrSz4 = a2;

return 2;

}

{

\*arrSz4 = a3;

return 3; // else a3 < a2

}

}

}

void populateArray(int \*arr, int aSz)

{

int theSize = aSz;

int theVal = 0;

for (int i = 0; i < theSize; i++)

{

theVal = rand() % MAXNUM; /\* code \*/

theVal++;

\*(arr + i) = theVal;

}

}

bool elementFoundInArray(int arr[], int aSz, int aVal)

{

int theSize = aSz;

int theVal = aVal;

bool found = false;

for (int i = 0; i < theSize && !found; i++)

{

if (theVal == arr[i])

{

found = true;

}

}

return found;

}

void displayArray(int \*arr, int aSz)

{

int theSize = aSz;

for (int i = 0; i < theSize; i++)

{

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

}

}

void sortTheArray(int \*arr, int aSz)

{

int theSize = aSz;

int refPosn = 0;

for (int i = 0; i < theSize - 1; i++)

{

refPosn = i;

for (int j = i + 1; j < theSize; j++)

{

if (\*(arr + refPosn) > \*(arr + j))

{

refPosn = j;

// printf("\n Ref posn , j Val : %d, %d", refPosn, j);

}

}

if (refPosn != i)

{

swap(arr, i, refPosn);

}

}

}

void swap(int \*arr, int posn1, int posn2)

{

int thePosn1, thePosn2, temp;

temp = \*(arr + posn1);

\*(arr + posn1) = \*(arr + posn2);

\*(arr + posn2) = temp;

}

**PS2-Week 1- 2 Qsn 13**

#include <stdio.h>

#include <stdlib.h>

#include <stdbool.h>

#include <time.h>

#define MAXNUM 25

// Windows: Shift + Alt + F - for formatting

void populateArray(int arr[], int arrSize);

void populateArrayRandom(int \*arr, int aSz);

int getKthLargest(int arr[], int arrSize, int aKno);

void printArray(int \*arr, int aSz);

void swap(int \*arr, int posn1, int posn2);

int main()

{

time\_t theSeed;

srand((unsigned)time(&theSeed));

int arrSize, refPosn, kNo;

printf("\nEnter number of integers you will input : ");

scanf("%d", &arrSize);

do

{

printf("\nEnter the 'K' number of the largest (a Value between 1 and %d ) : ", arrSize);

scanf("%d", &kNo);

} while ((kNo < 1) || (kNo > arrSize));

int arr[arrSize];

populateArrayRandom(arr, arrSize);

printArray(arr, arrSize);

int kthLargest = getKthLargest(arr, arrSize, kNo);

printf("\n");

printArray(arr, arrSize);

printf("\n%d %s Largest: %d", kNo, (kNo < 4 ? (kNo < 3 ? "nd" : "rd") : "th"), kthLargest);

printf("\n");

return 0;

}

void populateArray(int arr[], int arrSize)

{

printf("\nEnter array elements : \n");

for (int i = 0; i < arrSize; i++)

{

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

}

}

void populateArrayRandom(int \*arr, int aSz)

{

int theSize = aSz;

int theVal = 0;

for (int i = 0; i < theSize; i++)

{

theVal = rand() % MAXNUM;

theVal++;

\*(arr + i) = theVal;

}

}

int getKthLargest(int arr[], int arrSize, int aKno)

{

int kLargest;

int theKno = aKno;

int theSize = arrSize;

int theLgPosn = 0;

int theCnt = 0;

bool exch = false;

for (int i = 0; i < theSize && theCnt < theKno; i++)

{

exch = false;

theLgPosn = i;

for (int j = i + 1; j < arrSize; j++)

{

if (arr[theLgPosn] < arr[j])

{

theLgPosn = j;

exch = true;

}

}

if (exch)

{

swap(arr, i, theLgPosn);

theCnt++;

}

}

kLargest = arr[theKno - 1];

return kLargest;

}

void printArray(int \*arr, int aSz)

{

int theSize = aSz;

for (int i = 0; i < theSize; i++)

{

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

}

}

void swap(int \*arr, int posn1, int posn2)

{

int temp = \*(arr + posn1);

\*(arr + posn1) = \*(arr + posn2);

\*(arr + posn2) = temp;

}