Problems for Programming

Set C.1: 1D Arrays: *4 points per problem*

1. Write a program which accepts a sequence of numbers and calculates their sum and average. Ask the user to enter the number of values in the sequence.
2. Modify the program above, to read the data in from a file. Change the program, to use an array.
3. Form an ordered array and find out if there is a number *X*, and its position. Use linear (sequential) search.
4. Form an ordered array and find out if there is a number *X*, and its position. Use bisectional search.
5. Form an ordered array of at least 20 elements and find out if there is a number *X*, and its position. Use linear (sequential) search and bisectional search. Compare number of steps needed for each method.

Set C.2: 1D Arrays: *5 points per problem*

1. Write a program which sorts a randomly formed array by the bubble sort.
2. Write a program which sorts a randomly formed array by the selection sort.
3. Write a program which sorts a randomly formed array by the insertion sort.
4. *Implement any of the "fast sorting" algorithms. Find out how faster it works compared to C.2.1 – C.2.3.

Set D.1. 2D Arrays: *2 points per problem*

1. Describe a 2D array of $m \times n$, where m,n are constants.
2. Form a 2D array from the keyboard.
3. Form a 2D array of random values.
4. Write a Function (Procedure) which reads in a 2D array from a file.
5. Write a Function (Procedure) which prints out a 2D array on the screen.
6. Write a Function (Procedure) which prints out a 2D array into a file, with the file name being read in from the keyboard.

Set D.2-1 2D Arrays: *2 points per problem*

Form a 2D array of integer numbers. With use of function(s) or/and procedure(s):

1) Find the smallest positive element and its indices.
2) Find the average of all the elements, *x_avg*.
3) Find the average of all the negative elements, *x_neg_avg*.
4) Find the average of all the positive elements, *x_pos_avg*.
5) Find the product of *x_neg_avg* and *x_pos_avg*.
6) Find *[(x_pos_avg) div (x_avg)]·[(x_neg_avg) div (x_avg)]*
7) Compare *abs[(x_pos_avg) mod (x_avg)]* and *abs[(x_neg_avg) mod (x_avg)]]*
8) Find a product of the smallest positive element and the largest element.
9) Find a product of the smallest element and the largest element. Make sure that your product is of a proper data type.
10) Find a product of the smallest by its absolute value element and the largest by its absolute value element.

Set D.2- 2 2D Arrays: *3 points per problem*

1. Form a 2D array. Find the largest element and its indices.
2. Form a 2D array. Form a 1D array which elements are the sum of negative elements in rows of the 2D array. Form a 1D array which elements are the sum of positive elements greater than 10 in the columns of the 2D array.
3. Form a 2D array. Introduce a Boolean variable which would return "true" if there is a zero in the 2D array.
4. Form a square ($n \times n$) 2D array. Find out if the elements are symmetrically distributed along the main diagonal.
5. Form a 2D array. In each column, find the sum and number of elements that are: 1) divisible by *k1 or k2*; 2) belong to an interval *[a;b]*; 3) prime numbers; 4) positive and above the main diagonal.
6. Form a 2D array. Subtract the 2nd row from the first one. Add 3rd column multiplied by 2.5 to the 4th column.
7. * Form a 2D array of size 8×8, with the elements in the ascending order in odd rows, and in descending order in the even rows.
8. * Write a program that reads in from the keyboard coordinates of a queen and provides all the fields under control by the queen on the chess board.
   Hint: all the diagonals are either ascending, or descending. For each field on an ascending diagonal, $i + j = const = 2 \ldots 16$, and $i - j = const = -7 \ldots 7$ for each field on an descending diagonal.
9. * Write a program that reads in from the keyboard coordinates of a queen and a knight, and finds out if either one of them is endangered by the other.

Set D.2-3 2D Arrays: *3 points per problem*

Form a 2D array of real numbers, *A*. Use a *Procedure* and the *formatted output* to display the array on the screen.

Notation: $X^{<Y>}$ means *column* ***Y*** in *array* ***X***. (i.e., $A^{<i>}$ denotes elements of the i^{th} column of the array).

With use of function(s) or/and procedure(s) construct an array *B* in which:

1) 1st column is a product of $A^{<1>}$ and $A^{<2>}$.
2) 2nd column is a sum of $A^{<1>}, A^{<2>}, \ldots A^{<n>}$.
3) 3rd column is the absolute difference between $A^{<n-1>}$ and $A^{<n>}$.
4) 4th column is a negative of $(A^{<2>} + A^{<3>})$.
5) 5th column is $abs(A^{<2>} / A^{<1>})$,
6) 6th column is the $sqrt(abs(A^{<2>} / A^{<1>}))$
7) 7th column is the $A^{<2>} \cdot 5 - 3 \cdot A^{<3>}$,
8) 8th column is the $A^{<n>} \cdot 4.5 - 1.5 \cdot A^{<n-1>}$.

Set D.3-1 2D Arrays: *2 points per problem*

Form a 2D array of real numbers, *A*. With use of function(s) or/and procedure(s):

1) Calculate the determinant *det(A)*,
2) Calculate a matrix of the minors,
3) Calculate a matrix of the cofactors,
4) Calculate the transposed (*aka* adjugate *or* adjoint) matrix,
5) Calculate the inverse matrix (A^{-1}),
6) Calculate the determinant of the inversed matrix $det(A^{-1})$.

Set D.3-2 2D Arrays: *5 points per problem*

1. Write a program that solves a system of linear equations using the matrix method.
2. Write a program that solves a system of linear equations using the Cramer method.
3. Write a program that solves a system of linear equations using the Gauss method.

Set D.4 The Least Squares Method: *5 points per problem*

1. Write a program which uses the least-square-method to approximate experimental data points with the linear function.
2. *Write a program which uses the least-square-method to approximate experimental data points with the polynomial function.