ULAB

UNIVERSITY OF LIBERAL ARTS BANGLADESH

Course Title: Structured Programming

Course Code: CSE 1201 (Fall 2024)

Assignment

Submitted to:

Fatema Khan

Submitted by:

Student Name and ID

Md. Tazminur Rahman Tanim (242014124)

Department of CSE

University of Liberal Arts Bangladesh (ULAB)

Date : 29/12/2024

① Write a program in c to find the sum of all elements in an array.

Test Data:

Input the number of elements to be stored in array : 4

Input 4 elements in the array :

element - 0 : 10

element - 1 : 20

element - 2 : 30

element - 3 : 40

Expected output :

sum of all element in the array is : 100

Answer:

Algorithm:

① Initialize sum = 0

② Loop through each element in the array :

- Add the element to sum

③ Output the value of sum.

code:

```
#include <stdio.h>
int main () {
int n, i, sum = 0;
```

```
printf ("Input the number of elements to
        be stored in the array :  ");
scanf ("%d", &n);
int arr [n]
printf ("Input %d elements in the
                array :\n", n);
for (i=0; i<n ; i++){
printf ("element - %d : ", i);
scanf ("%d", & arr [i]);
sum += arr [i];
}
printf ("Sum of all elements in the
    array is : %d\n", sum);
    return 0;
}
```

Output result :

```
Input the number of element to be stored
 in the array : 4
Input 4 element in the array :
element -0 : 10
element - 1 : 20
element - 2 : 30
element - 3 : 40
sum of all element in the array is : 100
```

② Write a program in C to find the second largest element in an array. ed

Answer:

Alogorithm:

① Initialize firstLargest = secondLargest = INT_MIN

② Loop through each element:

- If the element is gather than firstLargest, update secondLargest = firstLargest and firstLargest = element.
- Else if the element is greater than secondLargest and not equal to firstLargest, update secondLargest = element

③ Output second Largest.

codes

```
#include <stdio.h>
#include <limits.h>

int main () {
int n, i, firstLargest = INT_MIN, second
    Largest = INT_MIN;

printf (" Input the number of elements
    to be stored in the array : ");
scanf ("%d", &n);
```

```
int arr [n]
printf ("Input %d elements in the array
        : \n ", n );

for (i = 0; i<n ; i++ ) {
printf ("element - %d : ", i );
scanf ("%d", & arr [i] );
if (arr [i] > firstLargest) {
secondLargest = firstLargest ;
firstLargest = arr [i] ;
}
else if
   (arr[i] > secondLargest && arr [i] !=
      firstLargest) {
secondLargest = arr[i];

}
}
printf ("Second largest element : %d \n",
        secondLargest );
return 0;
}
```

5

output result:

```
Input the number of elements to be stored
in the array : 5
Input 5 elements in the array :
element - 0 : 12
element - 1 : 35
element - 2 : 1
element - 3 : 10
element - 4 : 34
second largest element : 34
```

③ Write a program in c to separate odd and even integers into separate arrays.

Answer:

Algorithm:

① Initialize two arrays: odd[] and even[].

② Loop through each element in the input array:

- If the element is even, append it to even[].
- If the element is odd, append it to odd[].

③ Output even[] and odd[].

code:

```
#include <stdio.h>
int main() {
int n, i, evenIndex = 0, oddIndex = 0;
printf("Input the number of elements
    in the array : ");
scanf("%d", &n);

int arr[n], even[n], odd[n];
printf("Input %d elements in the array:
    \n", n);
```

```
for (i=0; i<n; i++){
printf("element - %d : ", i);
scanf("%d", &arr[i]);
if (arr[i] %2 == 0)
even [evenIndex++] = arr[i];
else
odd [oddIndex++] = arr[i];
}
printf("The even element are :\n");
for (i=0; i<evenIndex; i++) {
printf("%d", even[i]);
}
printf("\n The odd elements are :\n");
for (i=0; i<oddIndex; i++) {
    printf("%d", odd[i]);
}
printf("\n");
return 0;
}
```

(5)

output result:

Input the number of elements to be stored in the array : 5

Input 5 elements in the array:

element - 0 : 25

element - 1 : 47

element - 2 : 42

element - 3 : 56

element - 4 : 32

The Even element are:
42, 56, 32

The odd element are:
25, 47

④ Write a program in C to insert values in the array.

Answer:

Algorithm:

① Take input for the size, array elements, value to be inserted, and the position.

② Shift all elements from the position the right by one index.

③ Insert the value at the specified position.

④ output updated array.

code:

```
#include <stdio.h>
int main () {
int n, i, value, pos;
printf ("Input the size of array : ");
scanf ("%d", &n);

int arr [n+1]
printf ("Input %d elements in the array
    in ascending order : \n", n);

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

```
printf ("element - %d : ", i );
scanf ("%d", &arr[i]);
}
printf ("Input the value to be inserted:");
scanf ("%d", &pos);
for (i = n; i > pos; i--){
arr[i] = arr[i-1];
}
arr[pos] = value;
printf ("After Inserted the element the
        new list is :\n");
for (i = 0; i <= n; i++){
printf ("%d ", arr[i]);
}
printf ("\n");
return 0;
}
```

output result:

Input the size of array : 4

Input 4 element in the array in ascending order :

element - 0 : 1

element - 1 : 8

element - 2 : 7

element - 3 : 10

Input the value to be inserted : 5

Input the position where the value to be inserted : 2

After Insert the element the new list is :

1 , 8 , 5 7 10

⑤ Write a program in C for subtracting two matrices of the same size.

Answer:

Algorithm:

① Initialize a matrix result [ ] [ ] of the same size.

② Loop through each element of the matrices:

- compute result [i] [j] = matrix1 [i] [j] − matrix2 [i] [j].

③ Output result [ ] [ ].

Code:

```
#include <stdio.h>
int main ( ) {
int n, i, j ;
printf ("Input the size of the square
        matrix (less than 5) : ");
scanf ("%d", &n);
int matrix1 [n] [n], matrix2 [n] [n],
    result [n] [n];
printf ("Input element in the first
                matrix : \n");
  for (i=0 ; i<n ; i++) {
  for (j=0 ; j<n ; j++) {
```

```
        printf ("element - [%d] ; [%d] : ", i, j);
        scanf ("%d", &matrix1 [i] [j]);
    }
}
printf ("Input elements in the second
        matrix : \n");
for (i=0; i<n; i++) {
for (j=0; j<n; j++) {
printf ("element - [%d], [%d] : ", i, j);
scanf ("%d", &matrix2 [i] [j]);
}
}
printf ("The subtraction of two matrices
        is : \n");
for (i=0; i<n; i++) {
for (j=0; j<n; j++) {
result [i] [j] = matrix1 [i] [j] - matrix2 [i] [j];
printf ("%d ", result [i] [j]);
}
printf ("\n");
}
return 0;
}
```

output result:

```
Input the size of square matrix (less than 5)
: 2
Input elements in the the first matrix:
element - [0], [0] : 1
element - [0], [1] : 2
element - [1], [0] : 3
element - [1], [1] : 4

Input elements in the second matrix:
element - [0], [0] : 5
element - [0], [1] : 6
element - [1], [0] : 7
element - [1], [1] : 8

The subtraction of two matrices is:
-4  -4
-4  -4
```

⑥ Write a program in C to find the row with the maximum number of 1.

Answer:

Code:

```
#include <stdio.h>
  int main () {
  int rows, cols, i, J, maxOnes = 0;
    maxRowIndex = -1;
printf ("Enter the number of rows and columns:");
scanf ("%d %d", &rows, &cols);
  int array [rows] [cols];
printf ("Enter the elements of the 2D array
          : \n");
  for (i = 0; i < rows; i++) {
  for ( int count = 0;
  for (J = 0; J < cols; J++) {
  if (array [i] [j] == 1) {
        count++;
    }
  }
  if (count > maxOnes) {
    maxOnes = count;
    maxRowIndex = i;
  }
}
```

```
if (maxRowIndex != -1) {
printf("The Index of row with maximum 1s
        is : %d\n", maxRowIndex);
}
else {
printf("No row contains any 1s.\n");
}
return 0;
}
```

Output result:

```
Enter the number of rows and columns : 5 5

Enter the element of the 2D array :

0 1 0 1 1
1 1 1 1 1
1 0 0 1 0
0 0 0 0 0
1 0 0 0 1

The Index of row with maximum 1s is 1
```