CMPT 130: Lab Work Week 10

1. What is the output of the following code fragment? The code is assumed to be embedded in a correct and complete program.

```
int *a = new int[10];
int *p = a;
int i;
for (i = 0; i < 10; i++)`
   a[i] = i;
for (i = 0; i < 10; i++)
   cout << p[i] << " ";
cout << endl;
delete [] p;
```

2. What is the output of the following code fragment? The code is assumed to be embedded in a correct and complete program.

```
int array_size = 10;
int *a;
a = new int [array_size];
int *p = a;
int i;
for (i = 0; i < array_size; i++)
   a[i] = i;
p[0] = 10;
for (i = 0; i < array_size; i++)
   cout << a[i] << " ";
cout << endl;
delete [] a;
```

3. Analyze the following program and determine its output

```
void foo(int a, int& b, int* c, int* d)
{
   a += 5;
   b += 5;
   c += 5;
   *d += 5;
   return;
}
int main()
{
   //Create array
   int* A = new int[4];
   //Populate array
   for (int i = 0; i < 4; i++)
         A[i] = i+1;
   //Print array
   for (int i = 0; i < 4; i++)
         cout << A[i] << "\t";
   cout << endl;
```

```
    //Pass array elements to function
    foo(A[0], A[1], &(A[2]), &(A[3]));
    //Print array
    for (int i = 0; i < 4; i++)
          cout << A[i] << "\t";
    cout << endl;
    //Delete array
    delete[] A;

    system("Pause");
    return 0;
}
```

4. Write a program that creates a C++ dynamic array of integers of user desired size, populates the elements of the array with random integers in the range [0, 50] and then prints how many of them are even integers and how many are odd.
In this program, the array size must be read from the user as follows:

```
int arraySize;
cout << "Please enter the size of the array to be created: ";
cin >> arraySize;
```

5. Write a program that creates a C++ dynamic array of floats of user desired size, populates the elements of the array with random floats in the range [-1.0, 1.0) and then prints how many of them are positive floats and how many are negative floats.

6. Write a program that creates a C++ dynamic array of integers of user desired size, populates the elements of the array with random integers in the range [-10, 10] and then prints the sum of all the elements.

7. Write a program that creates a C++ dynamic array of integers of user desired size, populates the elements of the array with random integers in the range [-10, 10] and then prints the average of all the elements.

8. Write a program that creates a C++ dynamic array of integers of user desired size, populates the elements of the array with random integers in the range [-10, 10] and then prints the sum of the absolute values of all the elements.

9. Write a program that creates a C++ dynamic array of integers of user desired size, populates the elements of the array with random integers in the range [-10, 10] and then prints the absolute value of the sum of all the elements.

10. Write a program that creates a C++ dynamic array of integers of user desired size, populates the elements of the array with random integers in the range [-10, 10] and then prints the product of all the elements.

11. Write a program that creates a C++ dynamic array of integers of user desired size, populates the elements of the array with random integers in the range [-10, 10] and then prints the product of the absolute values of all the elements.

12. Write a program that creates a C++ dynamic array of integers of user desired size, populates the elements of the array with random integers in the range [-10, 10] and then prints the absolute value of the product of all the elements.

13. Write a program that creates a C++ dynamic array of integers of user desired size, populates the elements of the array with random integers in the range [-10, 10] and then prints the maximum and minimum elements of the array.

14. Write a function that takes a C++ dynamic array of integers and its size as arguments and returns the number of negative integers in the array. Test your function by writing a main program that creates a dynamic array of integers of user defined size, populates the array with random integers in the range [-5, 5] and calls the function to determine how many negative elements are there in the array and finally prints the message "A negative number is found in the array" if there is at least one negative element in the array; otherwise it prints the message "No negative number is found the array ".

15. Write a function that takes a C++ dynamic array of integers and its size as arguments and returns the number of prime numbers in the array. Assume all the elements of the array are positive integers greater than 1. Test your function by writing a main program that creates a dynamic array of integers of user defined size, populates the array with random integers in the range [2, 100] and calls the function to determine how many prime number elements are there in the array and finally prints the message "A prime number is found the array " if there is at least one prime number element in the array; otherwise it prints the message "No prime number is found the array ". You should write **`bool isPrime(int a)`** function to answer this question easily.

16. Write a function that takes a C++ dynamic array of characters and its size as arguments and returns the number of digits in the array. Test your function by writing a main program that creates a dynamic array of characters of user defined size, populates the array with random characters whose ascii codes are in the range [48, 122] and calls the function to determine how many digit character elements are there in the array and finally prints the message "A digit is found in the array " if there is at least one digit character element in the array; otherwise it prints the message "No digit character is found the array ".

17. Write a function that takes a C++ dynamic array of characters and its size as arguments and returns the number of alphabets in the array. Alphabet means any upper case or lower case English letter. Test your function by writing a main program that creates a dynamic array of characters of user defined size, populates the array with random characters whose ascii codes are in the range [48, 122] and calls the function to determine how many alphabet character elements are there in the array and finally prints the message "An alphabet is found the array " if there is at least one alphabet character element in the array; otherwise it prints the message "No alphabet character is found the array".

18. Write a function that takes two dynamic arrays of integers A and B and their sizes sizeA and sizeB respectively and returns a new dynamic array of integers of size sizeA+sizeB and whose elements those elements of A followed by those elements of B.

19. Write a function that takes a dynamic array of floats and its size and returns a new array of floats whose elements are the elements of the argument array in reverse order.

20. Write a function named **`isFound`** that takes a dynamic array of integers, its size and an integer number as arguments and that returns true if the integer number is found in the array otherwise it returns false.

21. Write a function named **`countElements`** that takes two dynamic arrays A and B of integers and their sizes as arguments and returns the number of elements A that are found in B. That is this function returns how many elements of A are found in B. Use your **`isFound`** function defined above.

22. Write a function that takes two dynamic arrays A and B of characters and their sizes as arguments and returns true if every element of A is found in B; otherwise returns false. Use your **`countElements`** function defined above.

23. Write a function that takes two dynamic arrays of integers and their sizes and returns true if the two arrays are identical otherwise returns false. Two arrays are identical if both arrays have the same size and that they contain the same elements in the same order.

24. Write a function that takes two dynamic arrays A and B of integers and their sizes as arguments and returns true if every element of A is found in B and that every element of B is found in A; otherwise returns false. Please note that the arrays may contain duplicated elements. Use your **`isFound`** and **`countElements`** functions defined above.

25. Write a function that takes two dynamic arrays of integers A and B and their sizes as arguments and returns a new dynamic array of integers whose elements are the elements of A that are found in B. Note that this function must return a dynamic array.

In your test main program, you will notice that the main program will not know how many elements the returned dynamic array has. This is because we only returned the dynamic array from the function but not its size. Moreover we can't return both the dynamic array and its size because a function can return only one value. So what should we do? Well we should declare the size of the array in the main program and pass it by reference to the function and then the function must assign it the size of the dynamic array created in the function. In conclusion we see that the function must take FIVE arguments; namely array A, its size sizeA, array B, its size sizeB, and integer argument size (by reference). Then this function must

- Count how many elements of A are found B.
- Assign the count to the parameter size.
- Create a new dynamic array of integers of the size just computed.
- Fill the array with elements of A that are found in B.
- Return the dynamic array.

26. Analyze the following complete C++ program carefully and determine what the output of the program is. You should not type the code in a computer for otherwise you won't learn anything. First carefully analyze what each function does and then analyze the main program.

```
#include <iostream>
using namespace std;

void printArray(const int *arr, const int arr_size)
{
    for (int i = 0; i < arr_size; i++)
            cout << "Element at " << i << " = " << arr[i] << endl;
}

int findElementIndex(const int *arr, const int arr_size, const int e)
{
    for (int i = 0; i < arr_size; i++)
            if (arr[i] == e)
                    return i;
    return -1;
}

void deleteElement(int* &arr, int &arr_size, const int e)
{
```

```
    int index = findElementIndex(arr, arr_size, e);
    if (index != -1)
    {
            int *B = new int[arr_size-1];
            for (int i = 0; i < index; i++)
                B[i] = arr[i];
            for (int i = index+1; i < arr_size; i++)
                B[i-1] = arr[i];
            delete [] arr;
            arr_size--;
            arr = B;
    }
    return;
}
int main()
{
    //Create a dynamic array
    int size = 8;
    int *A = new int[size];
    for (int i = 0; i < size; i++)
            A[i] = 15+i;

    //Print the dynamic array
    cout << "Originally the array is..." << endl;
    printArray(A, size);

    //Call deleteElement function to delete an element whose value is 12
    int x = 12;
    deleteElement(A, size, x);
    cout << "After calling the deleteElement function with x = 12, the array is..." << endl;
    printArray(A, size);

    //Call deleteElement function to delete an element whose value is 18
    x = 18;
    deleteElement(A, size, x);
    cout << "After calling the deleteElement function again with x = 18, the array is..." << endl;
    printArray(A, size);

    delete [] A;

    system("pause");
    return 0;
}
```

27. One problem with dynamic arrays is that once the array is created using the new operator then we can't shrink or expand the allocated memory. But what if we want to append (that is to insert one more element at the end of the array) which needs the allocated memory to be expanded?

Write a function named **appendElement** that takes a dynamic array, its size and an integer value as arguments and that appends the integer argument to the dynamic array. The function declaration is provided below.

```
void appendElement(int* &arr, int &arr_size, const int e)
```

Now, use the following program to test your function.

```
int main()
{
```

```
    int size = 0;
    int *A;
    for (int i = 0; i < 5; i++)
    {
        appendElement(A, size, 15+i);
        for (int i = 0; i < size; i++)
            cout << A[i] << "  ";
        cout << endl;
    }
    delete [] A;
    system("pause");
    return 0;
}
```

The output should be as follows

```
15
15    16
15    16    17
15    16    17    18
15    16    17    18    19
```

Multi-Dimensional Arrays

28. Consider the following code fragment that prints a 2D array (matrix). Assume the matrix has been already created and populated and that it has R rows and C columns.

```
for (int i = 0; i < R; i++)
{
    for (int j = 0; j < C; j++)
    {
        cout << M[i][j] << "\t";
    }
    cout << endl;
}
```

- Re-write the given code fragment so that the printing is done using indexing on index `i` but no indexing on index `j`. The `j` index rather should use a de-referencing.
- Re-write the given code fragment so that the printing is done using indexing on index `j` but no indexing on index `i`. The `i` index rather should use a de-referencing.
- Re-write the given code fragment so that the printing is done without indexing. That is both index `i` and index `j` should use dereferencing.

29. Write a function that takes a matrix, its column size and an index **r** and returns the sum of the elements of the matrix at row **r**.

30. Write a function that takes a matrix, its row size and an index **c** and returns the sum of the elements of the matrix at column **c**.

31. Write a function that takes a matrix, its column size, and two indexes **r1** and **r2** and swaps the row of the matrix at row index **r1** with the row of the matrix at row index **r2**.

32. Write a function that takes a matrix, its row size, and two indexes **c1** and **c2** and swaps the column of the matrix at column index **c1** with the column of the matrix at column index **c2**.

33. Write a function that takes a matrix and its sizes and returns true if all the rows of the matrix have the same sum and returns false otherwise.

34. Write a function that takes a matrix and its sizes and returns true if all the columns of the matrix have the same sum and returns false otherwise.

35. Write a C++ function that takes a two dimensional dynamic array (matrix) and its sizes and returns true if the matrix is an identity matrix and returns false otherwise. Remark: a matrix is an identity matrix if it is a square matrix (that is **row size = column size**) and all its diagonal elements are exactly equal to 1 while every other element is 0.

36. Write a C++ function that takes a two dimensional dynamic array (matrix) and its sizes and returns true if the matrix is a lower matrix and returns false otherwise. Remark: A matrix M is lower matrix if it is a square matrix (row size = column size) and that all the elements of the matrix above the main diagonal are zero. Other elements don't matter.

37. Write a C++ function that takes a two dimensional dynamic array (matrix) and its sizes and returns true if the matrix is an upper matrix and returns false otherwise. Remark: A matrix M is an upper matrix if it is a square matrix (row size = column size) and that all the elements of the matrix below the main diagonal are zero. Other elements don't matter.

38. Write a function named **`dotProduct`** that takes two same size dynamic arrays **A** and **B** of integers and their size as arguments and returns the dot product of the dynamic arrays. Remark:- The dot product of two vectors (that is arrays) **A** and **B** of equal length **size** is given by

$$dotProduct(A, B) = \sum_{i=0}^{size-1} A[i] * B[i]$$

39. Write a function that takes two matrices M1 and M2 and their sizes and returns the product of the matrices. Make sure you first understand the mathematics of matrix multiplication before wasting time trying to write the code. Two matrices are compatible for multiplication if the number of columns of M1 is equal to the number of rows of M2. Otherwise they are not compatible for multiplication and your function should return empty matrix. Use the **`dotProduct`** function when you answer this question.

40. Write a function that takes a matrix and its row and column sizes and returns the transpose of the matrix.