|  |
| --- |
| **Assignment # 2**  ***Session*: Spring 2022 *Total marks*: 100**  ***Name*** :***NIMRA MAQBOOL Roll number* : BSCE21012** |

***Submission:***

• *Email instructor or TA if there are any questions. You cannot look at others’ solutions or use others’ solutions, however, you can discuss it with each other. Plagiarism will be dealt according to the course policy.*

*• Submission after due time will not be accepted.*

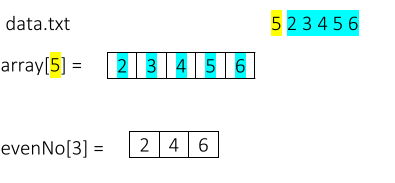
**There should be a Report explaining your code and highlighting results. Follow this naming convention for your report RollNumber\_Assignment#.pdf e.g BSCE21001\_Assignment2.pdf.**

**TASK: 1**

You are given a file containing integers. The first integer is representing the size of the array (i.e. numbers present in the file). You are required to read the first integer (size), allocate memory dynamically of that size, and read the remaining integers in the array.

Now, Find the even numbers from the above-filled array and store them in array evenNo[]. Make sure we do not know how many even numbers are present in the array, so we need to use regrow to make memory efficient program.

Example:



Prototype of function will be:

int\* findEven(string fileName, int input, int size, int count);

|  |
| --- |
| **FUNCTION.CPP**  int \*findEven(string fileName, int input, int size, int count) {  fstream myFile;  myFile.open(fileName);  if (!myFile.is\_open()) {  cout << "the file is not open ";  } else {  myFile >> input;  cout << "THE SIZE OF ARRAY = " << input << endl;  int \*array = new int[input];  int \*even = new int[input];   for (int i = 0; i < input; i++) {  myFile >> array[i];  }  for (int i = 0; i < input; i++) {  if (array[i] % 2 == 0) {  count++;  }  }  cout << "count =" << count << endl;  cout << "EVEN ARRAY is= ";  int j = 0;  for (int i = 0; i < input; i++) {  if (array[i] % 2 == 0) {  even[j] = array[i];  cout << even[j] << " ";  j++;  }  }  myFile.close();  delete[]array;  return even;  } }  **MAIN.CPP**  int input; int size; int count; string fileName; cout << "please enter file name" << endl; getline(cin, fileName); int \*evenNumbers = new int[size]; evenNumbers = findEven(fileName, input, size, count); findEven(fileName, input, size, count);  **OUTPUT:**  **Graphical user interface, text  Description automatically generated** |

**TASK: 2**

You are given a file containing character’s pairs. Write a function that finds and stores pairs in an array (say sameCharacters[]), if both the characters (of a pair) are the same. Make sure we don’t know how many pairs are there in the file. Use 1D arrays.

**Data.txt**

**Ab ac aa be kk ik dk le on of th ir mm is at no**

**Expected**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **a** | **a** | **k** | **k** | **m** | **m** |

**Output: sameCharacters for given data would be: sameCharacters[6]**

**Note: initially size of sameCharacters[] will be 1. We are required to use regrow here.**

Prototype of function will be:

char\* read(string fileName, int &size, int &countOfPairs, char \*ptr);

|  |
| --- |
| **FUNCTION.CPP:**  char \*read(string fileName, int &size, int &countOfPairs, char \*ptr) {  string line;  size = 1;  int count = 0;  ifstream file;  file.open(fileName);  if (!file.is\_open()) {  cout << "the file is not open" << endl;  } else {  getline(file, line);  size = line.size();  cout << "size =" << size << endl;  int j = 0;  for (int i = 0; i < size; i++) { // cout << line[i] << " ";  if (line[i] == line[i + 1]) {  countOfPairs++;  }  }  cout << "countOfPairs = " << countOfPairs++ << endl;  char \*sameCharacter = new char[countOfPairs]; // int j=0;  for (int i = 0; i < size;) {  if (line[i] == line[i + 1]) {  sameCharacter[j] = line[i];  sameCharacter[j + 1] = line[i];  cout << sameCharacter[j] << sameCharacter[j + 1] << " ";   i++;  j += 2;  } else {  i++;  }  }  return sameCharacter;  }  **MAIN.CPP:**  char \*ptr; int size; int countOfPairs; string fileName; cout << "please enter file name" << endl; getline(cin, fileName); read(fileName, size, countOfPairs, ptr); return 0;  **OUTPUT:**  **A picture containing text, orange, dark, close  Description automatically generated**  **Text  Description automatically generated** |