|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **National University of Computer and Emerging Sciences, Lahore Campus** | | | | |
|  | **Course Name:** | **Programming Fundamentals** | **Submission date** |  |
| **Assignment** | **2** | **Total Marks** | **80** |
| **Section** | **BDS-1B & BDS-1D** | **Weight** | **5** |

**SUBMIT EACH QUESTION AS A SEPARATE CPP FILE WITH YOUR ROLL NUMBER AND QUESTION NUMBER (21L-1111\_Q1.cpp). IF THIS FORMAT IS NOT FOLLOWED, YOUR SUBMISSION WILL NOT BE MARKED.**

**Follow the good programming practices; properly indent your code, choose meaningful variable and function names, and add comments where necessary. You may not use any predefined functions.**

**Question 1 [marks = 5 + 5]**

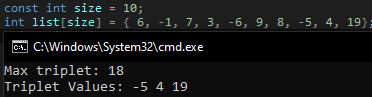
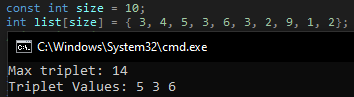
**a)**

**Consider a 1D integer array of size 10 that has unsorted values. You must write a c++ program that finds the maximum triplet sum i.e. the maximum sum obtained by any 3 consecutive values.**

**For example**

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Array** | **3** | **4** | **5** | **3** | **6** | **3** | **2** | **9** | **1** | **2** |

**this array has triplet sums (3+4+5=12, 4+5+3=12, 5+3+6=14, 3+6+3=12, …). The maximum triplet sum is 5+3+6=14.**

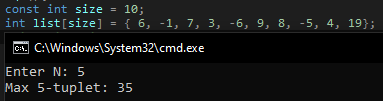
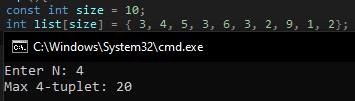
**Your c++ program must output this maximum sum AND the three values that are part of the triplet. You do not need to take array values as input, simply hardcode any random values.**

**b)**

**The concept of a max triplet can be generalized to a max n-tuplet where n can be 1, 2, 3, … up until the size of the array.**

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Array** | **3** | **4** | **5** | **3** | **6** | **3** | **2** | **9** | **1** | **2** |

**This array has 4-tuplet sums (3+4+5+3=15, 4+5+3+6=18, …). The maximum 4-tuplet sum is 20 (6+3+2+9). Similarly the max 5-tuplet sum is 23 (3+6+3+2+9).**

**In the same file as Q1(a) write additional code that takes N as input from user and then finds the max n-tuplet from the same array you have used in Q1(a). In this part you do not need to display the n values that make up the n-tuplet.**

**Question 2 [marks = 20]**

**The c++ int datatype can only store a limited range of numbers, but in many calculations we have to deal with large numbers. Instead of using the c++ datatypes, we can store very large numbers as an array of digits. We can call this array a BigInt. We must then also be able to perform the following basic operations on these BigInts.**

* **bigIn()**

This function takes a BigInt array and its size and inputs the digits of the BigInt from the console.

* **bigOut()**

This function takes a BigInt array and its size and outputs the BigInt onto the console. Leading 0s **must not** be printed (0005 should be printed as 5, not 0005).

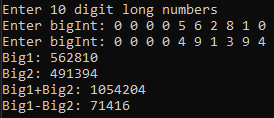
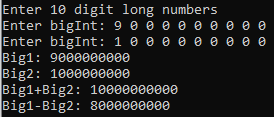
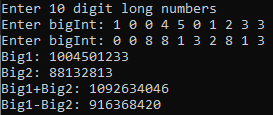
* **bigAdd()**

This function takes 3 BigInts A, B and C as well as their size. A and B are the same size arrays but C has 1 more element than A and B (number of digits can increase by addition). The function then adds A B and stores the result in C.

* **bigDiff()**

This function takes 3 BigInts A, B and C as well as their size. A, B, and C are the same size. The function then subtracts B from A (A – B) and stores the result in C. Assume that the answer will never be negative and A will always be greater than B.

**You must use the given main function. DO NOT make any changes to this main function. You must implement all the functions called in main. You may create additional functions for your convenience. You will not receive any marks if you convert the BigInt into a normal int to perform operations.**



int main(){

const int size = 10;

int big1[size];

int big2[size];

int big3[size + 1];

int big4[size];

cout << "Enter " << size << " digit long numbers\n";

bigIn(big1, size);

bigIn(big2, size);

cout << "Big1: ";

bigOut(big1, size);

cout << endl;

cout << "Big2: ";

bigOut(big2, size);

cout << endl;

bigAdd(big1, big2, big3, size);

cout << "Big1+Big2: ";

bigOut(big3, size+1);

cout << endl;

bigDiff(big1, big2, big4, size);

cout << "Big1-Big2: ";

bigOut(big4, size);

cout << endl;

return 0;

}

**Hint**

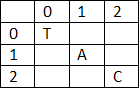
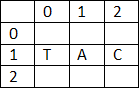
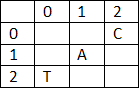
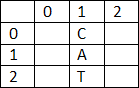
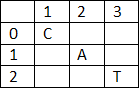
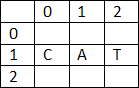
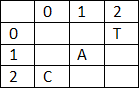
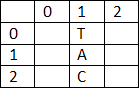
You may find it easier to implement the operations if you store the digits in reverse. For example storing 37581 as

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | 0 | 1 | 2 | 3 | 4 |
| Big1 | 1 | 8 | 5 | 7 | 3 |

**QUESTION 3 [Marks = 30]**

A word search puzzle is a popular kids puzzle which involves a grid of randomized alphabets. A list of words is also included and the child has to find those words in the grid of alphabets. These words can appear horizontally, vertically or diagonally.

Your task is to create a cpp program that solves the puzzle described above. You are provided a *main\_q3.cpp* with an already loaded grid and word list. You must search the grid for all words in the list; each word will appear only once. You must output the found word, along with the index (i,j) it was found at as well as the direction it was found in. There can be 8 possible directions a word can appear in and each direction is given a one digit code as shown below.



DIRECTION 0

Bottom to Top

DIRECTION 1

Bottom Left to Top Right

DIRECTION 2

Left to Right

DIRECTION 3

Top Left to Bottom Right

DIRECTION 4

Top to Bottom

DIRECTION 5

Top Right to Bottom Left

DIRECTION 6

Right to Left

DIRECTION 7

Bottom Right to Top Left

A sample word search is given below:

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **0** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** |
| **0** | T | N | E | M | N | G | I | S | S | A |
| **1** | B | N | C | A | O | M | P | J | W | R |
| **2** | C | L | A | R | I | F | Y | H | X | R |
| **3** | L | O | S | C | T | G | H | C | E | V |
| **4** | A | N | M | H | S | Y | S | T | E | M |
| **5** | S | T | I | P | E | Q | N | A | F | E |
| **6** | S | S | E | Q | U | E | N | C | E | M |
| **7** | U | E | F | N | Q | T | G | Q | W | O |
| **8** | D | K | R | O | W | T | E | N | K | R |
| **9** | A | O | M | O | D | N | A | R | T | Y |

WORDS: COMPUTER, QUESTION, CLARIFY, … .

Expected Output:

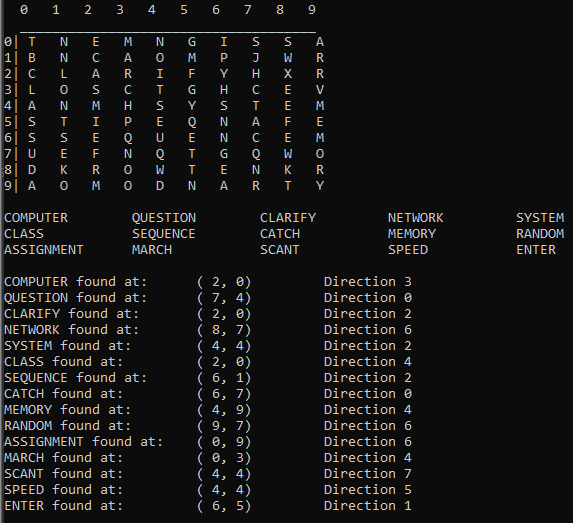
COMPUTER found at ( 2, 0) Direction 3

QUESTION found at ( 7, 4) Direction 0

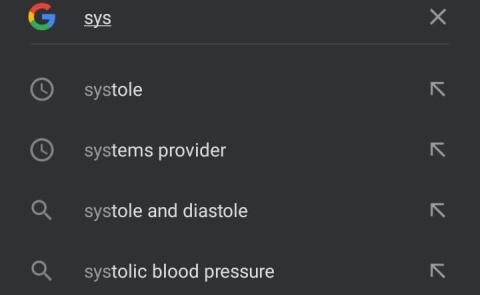
CLARIFY found at (2, 0) Direction 2

Your program must work for a grid of size 10x10 and a list of 15 words. You are provided with a grid and word list to verify your code, but your submissions will be tested on another sample grid so make sure your code is generic and can work on any grid.

You **ARE NOT ALLOWED to use any predefined functions** . String functions must be implemented by yourself.

**SAMPLE OUTPUT (**your output can look different however the found word, the position at which it is found, and the direction must be output on to the console**)**

**QUESTION 4[Marks = 20]**

Predictive text has become a very important tool in most modern devices. It allows you to see a number of possible words while you are in the middle of typing the actual word.

For Example:

Your task is to create a cpp program that will implement the functionality of a predictive text software. You will take input from the user character by character, and for each character you will suggest what word the user is trying to type. You are given a *main\_q4.cpp* file that contains an array of words from which you will offer predictions to the user.

So far when you have taken input from the user in c++, the user has to press ENTER. This does not make much sense for a predictive text program. Fortunately there is another way to take input from the user without having to press the ENTER key repeatedly. This makes use of the **getch()** function found in the **<conio.h>** library. The following code will continuously take input from user. If enter is pressed (ASCII = 13) the program will clear the screen, and if ‘0’ is pressed the loop will be exited and the program will terminate. (Give it a try before attempting the question).

#include <iostream>

#include <conio.h>

using namespace std;

int main(){

char a = 0;

while(true){

a = getch();

cout << a << ' ';

if (a == 13){//if ENTER pressed

system("cls");

cout << "Enter pressed" << endl;

}

else if (a == '0')

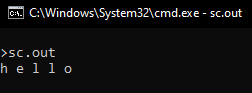
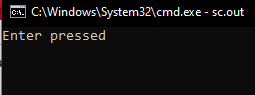
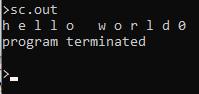
break;

}

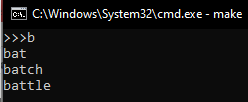
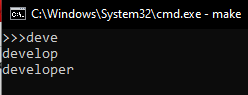
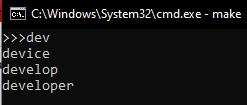
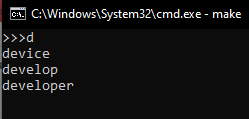
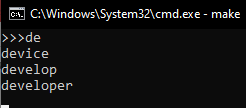
cout << endl << "program terminated" << endl;

return 0;

}



Study the sample code above and use the *main\_q4.cp* that contains an array of 20 words**.** Then using **getch()** take key inputs from the user and for each input, output all the possible words from the word array that the user may type. You MAY NOT use built in string functions. All string functions must be implemented yourself. Like in the sample code, terminate the program if the user enters ‘0’. If the user presses ENTER, clear the screen and start making new predictions according to the characters entered.



**<-------- d pressed, words starting with d displayed**

**<-------- ENTER pressed, now new predictions will be made**

**<-------- b entered**

**<-------- o entered, no words found that start with bo**

**<-------- 0 entered, loop exited, input terminated**

