Department of Computer Science

COS132 2015
Practical 5

Mr Madoda Nxumalo

UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

0 Introduction

0.1 Uploads and Deadline

This practical is a 3-hour in-lab practical, and as such students must write it at the Informatorium during the normal practical session hours. The uploads will become available at practical session hours as of **17 April - 23 April**.

Make sure that you have submitted your code to Fitchfork all your tasks at the end of the practical session. **No late submissions will be accepted**.

0.2 Plagiarism Policy

All submissions for this practical and all future practical should be documented as specified in our plagiarism policy. Refer to the notes about our plagiarism policy which can be found on the COS132 website on ClickUP.

1 Vectors

You have been introduced to vectors at some previous level of mathematics. At this level you must be familiar with basic operations on vectors. They can be added, subtracted, multiplied by a scalar and have a dot product etc. Vectors are represented as a column matrix as shown below.

$$\begin{pmatrix} 2 \\ -9 \end{pmatrix} \quad \begin{pmatrix} 1 \\ -2 \\ 3 \end{pmatrix} \quad \begin{pmatrix} 1 \\ -2 \\ -3 \end{pmatrix} \quad \begin{pmatrix} 1 \\ 2 \\ -3 \\ 4 \end{pmatrix}$$

In this in-lab practical session we want to implement a a set of functions for vectors with integer elements. We store the vector elements in arrays. In general, the array $\mathtt{v} = \{\mathtt{x}_0, \mathtt{x}_1, \ldots, \mathtt{x}_\mathtt{n}\}$ stores the elements for the vector;

$$v^{-} = \begin{pmatrix} x_0 \\ \vdots \\ x_n \end{pmatrix}$$

1.1 Functions

You must implement these functions prototypes that relates to vectors using arrays to store the column vector integers.

void `printVector(`**const int** `v[], `**size_t** `s)` - print the elements of a vector to `cout` using this general format: $[v_1, v_2, \ldots, v_3]$. For example, if

$$v = \begin{pmatrix} 1 \\ -2 \\ -3 \end{pmatrix}$$

then `printVector(v, 3)` outputs `[ 1, -2, -3 ]`

`printVector` should print out angular brackets (not preceded or followed by any spaces). `printVector` must not output any linebreaks. Also note that the final element should not have a comma after it.

It is important that you implement `printVector` correctly before implementing the other functions. Your implementation of `printVector` will be used to assess the correctness of the following functions.

void `addVectors(`**const int** `v1[], `**const int** `v2[], `**int** `r[], `**size_t** `dims)` - this function adds corresponding indices of the two vectors `v1` and `v2`. The result is stored in vector `r`. v1 and v2 must have the same number of dimensions (given by `dims`).

void `subtractVectors(`**const int** `v1[], `**const int** `v2[], `**int** `r[], `**size_t** `dims)` - this function subtracts corresponding indices of vector `v2` from `v1` and the result is stored in vector `r`. That is, `r[i] = v1[i] - v2[i]`. v1 and v2 must have the same number of dimensions (given by `dims`).

`double` `magnitude(`**`const int`** `v[],` **`size_t`** `dims)` calculates the magnitude of vector v, as follows:

$$magnitude = \sqrt{(x_0^2 + \ldots + x_n^2)}$$

for a given vector

$$v^{-} = \begin{pmatrix} x_0 \\ \vdots \\ x_n \end{pmatrix}$$

where n is the dimensions of the vector.

`bool` `equalVectors(`**`const int`** `v1[],` **`const int`** `v2 [],` **`size_t`** `dims)` - returns `true` if two vectors have equal corresponding values at all indices, otherwise returns `false`.

`int` `dotProduct(`**`const int`** `v1[],` **`const int`** `v2[],` **`size_t`** `dims)` - returns the dot product $v1 \bullet v2$ for two vectors with an equal number of dimensions. This is calculated as the sum of product of corresponding elements. For example, if:

$$v1 = \begin{pmatrix} 5 \\ -2 \\ 3 \end{pmatrix} \quad v2 = \begin{pmatrix} 1 \\ -2 \\ -3 \end{pmatrix}$$

then the dot product of v1 and v2 is $(5 \times 1) + (-2 \times -2) + (3 \times -3) = 0$

1.2 Sample Output

The user is expected to input the number of dimensions of a vector and then input the elements of the vector. Here is a sample output with all input values are written in bold-font.

```
Enter the number of dimensions of the vectors: 4
Enter the elements of the vector p: 2 3 4 5
Enter the elements of the vector q: 4 -4 3 7
The sum of vectors p and q is: [ 6, -1, 7, 12 ]
The difference of vectors; q from p is: [-2, 7, 1, -2 ]
The magnitude of vector p is: 7.348
The vectors p and q are: NOT EQUAL
The dot product of p is: 43
```

1.2.1 Comments

- Your main file and makefile will not be tested. However, this given test suite does not have full test coverage. You are encouraged to test your program with more test cases.
- **size_t** is an alias of one of the fundamental unsigned integer types, defined in `<cstddef>`. It is a type that is guaranteed to be able to represent any array index, or the size of any object in bytes. **size_t** is the type returned by the `sizeof` operator and is widely used in the standard library to represent sizes and counts. You should *prefer* using **`size_t`** instead of **`int`** or **`unsigned`** for variables that hold the size of an array, and for loop counters that represent array indexes. For example:

```
    for (size_t i=0; i<len; i++)
        array[i] = ...;
```

1.3 Submission

There are multiple upload slots for this session. Each upload slot focuses on testing only one of the functions. In each case you have to submit the following files:

- **vectors.h**
- **vectors.cpp**

The tarball containing the above mentioned files can be created by the following command.

```
tar -cvz vectors.h vectors.cpp -f vectors.tar.gz
```

As usual, the above tar command should be reissued before uploading an improved version of `vectors.cpp` or `vectors.h`

1.4 Marks Allocation

Task	**Percentage**	**Maximum mark**
Add Vector	13%	4
Dot Product	20%	6
Equal Vector	17%	5
Magnitude	17%	5
Print Vector	20%	6
Subtract	13%	4
Total	**100**	**30**