# Group 3: Nguyễn Đức Duy Nguyễn Công Hoàng Trần Quốc Bảo

Project 2

## Checkpoint

#### 1. Assume pint is a pointer variable. Is each of the following statements valid or invalid? If any is invalid, why?

A) pint++; // Valid

B) -–pint; // Valid

C) pint /= 2; // Invalid. Pointer to an address cannot be diveded

D) pint \*= 4; // Invalid. Pointer to an address cannot be multiplied

E) pint += 2; // Valid

#### 2. Is each of the following definitions valid or invalid? If any is invalid, why?

A) int ivar;

int \*iptr = &ivar;

Valid

B) int ivar, \*iptr = &ivar;

Valid

C) float fvar;

int \*iptr = &fvar;

Invalid. Pointer to an int can’t point to a float

D) int nums[50], \*iptr = nums;

Valid

E) int \*iptr = &ivar;

int ivar;

Invalid. Ivar is declared after pointer. Pointer is pointer to an un declared variable.

## Programming Challenges

#### 1.

#### Array Allocator (1)

#### Write a function that dynamically allocates an array of integers. The function should accept an integer argument indicating the number of elements to allocate. The function should return a pointer to the array.

#### Test Scores #1 (2)

#### Write a program that dynamically allocates an array large enough to hold a userdefined number of test scores. Once all the scores are entered, the array should be passed to a function that sorts them in ascending order. Another function should be called that calculates the average score. The program should display the sorted list of scores and averages with appropriate headings. Use pointer notation rather than array

#### notation whenever possible.

#### Input Validation: Do not accept negative numbers for test scores.

#### getString Function (5)

#### Write a function named getString that has a local char array of 80 elements. The function should ask the user to enter a sentence, and store the sentence in the array. Then the function should dynamically allocate a char array just large enough to hold the sentence, plus the null terminator. It should copy the sentence to the dynamically allocated array, and then return a pointer to the array. Demonstrate the function in a complete program.

**#include <iostream>**

**using namespace** std;

**int** \*allocIntArr(**int** size, **int** min, **int** max) {

**int** \*dest = **new int**[size];

**for** (**int** i = 0; i < size; ++i) {

**do** {

cout << "Enter element " << i

<< " for integers array:\t";

cin >> dest[i];

**if** (dest[i] < min || dest[i] > max)

cout << "Invalid data. Try again.";

} **while** (dest[i] < min || dest[i] > max);

}

**return** dest;

}

**int** \*allocIntArr(**int** size) {

**return** allocIntArr(size, 1 << 31, ~(1 << 31));

}

**void** selSort(**int** \*list, **int** size) {

**for** (**int** i = 0; i < size; ++i) {

**int** min = list[i];

**int** index = i;

**for** (**int** j = i; j < size; ++j) {

**if** (list[j] < min) {

min = list[j];

index = j;

}

}

list[index] = list[i];

list[i] = min;

}

}

**double** getAvg(**const int** \*list, **int** size) {

**double** temp = 0.0;

**for** (**int** i = 0; i < size; ++i) {

temp += list[i];

}

**return** temp / size;

}

**char** \*getString() {

**const int** LEN = 80;

**char** sent[LEN] = {0}; // Bunch of zeros

cout << "Enter a sentence:\t";

cin.ignore(256, '\n');

cin.getline(sent, 80);

**char** \*res = **new char**[LEN];

**for** (**int** i = 0; i < LEN; ++i) {

res[i] = sent[i];

**if** (sent[i] == '\0')

**break**;

}

**return** res;

}

**int** main() {

cout << "=====================\n";

**int** size;

cout << "How many elements do you want?:\t";

**do** {

cin >> size;

**if** (size < 0)

cout << "That must be > 0. Try again.";

} **while** (size < 0);

**int** \*l = allocIntArr(size);

cout << "\nYou just entered a list of: ";

**for** (**int** i = 0; i < size; ++i) {

cout << l[i] << ", ";

}

cout << endl;

**delete**[] l;

cout << "=====================\n";

cout << "How many tests are there?\t";

cin >> size;

**int** \*tests = allocIntArr(size, 0, 100);

selSort(tests, size);

cout << "\n\nYour test scores are:\t";

**for** (**int** j = 0; j < size; ++j) {

cout << tests[j] << ", ";

}

cout << "\nYour average score is: "

<< getAvg(tests, size);

cout << endl;

**delete**[] tests;

cout << "=====================\n";

cout << "\nEnter some sentence.\n";

**char** \*sent = getString();

cout << "\n\nYou just entered:\t"

<< sent << endl;

**delete**[] sent;

cout << "=====================\n";

**return** 0;

}

#### 2.

#### Mode Function (8)

#### In statistics, the mode of a set of values is the value that occurs most often or with the greatest frequency. Write a function that accepts as arguments the following:

#### A) An array of integers

#### B) An integer that indicates the number of elements in the array

#### The function should determine the mode of the array. That is, it should determine which value in the array occurs most often. The mode is the value the function should return. If the array has no mode (none of the values occur more than once), the function should return -1. (Assume the array will always contain nonnegative values.)

#### Demonstrate your pointer prowess by using pointer notation instead of array notation in this function.

#### Median Function (9)

#### In statistics, when a set of values is sorted in ascending or descending order, its median is the middle value. If the set contains an even number of values, the median is the mean, or average, of the two middle values. Write a function that accepts as arguments the following:

#### A) An array of integers

#### B) An integer that indicates the number of elements in the array

#### The function should determine the median of the array. This value should be returned as a double . (Assume the values in the array are already sorted.)

#### Demonstrate your pointer prowess by using pointer notation instead of array notation in this function.

#### Movie Statistics (14)

#### Write a program that can be used to gather statistical data about the number of movies college students see in a month. The program should perform the following steps:

#### A) Ask the user how many students were surveyed. An array of integers with this many elements should then be dynamically allocated.

#### B) Allow the user to enter the number of movies each student saw into the array.

#### C) Calculate and display the average, median, and mode of the values entered. (Use the functions you wrote in Problems 8 and 9 to calculate the median and mode.)

#### Input Validation: Do not accept negative numbers for input.

**#include <iostream>**

**using namespace** std;

**void** selSort(**int** \*list, **int** size) {

**for** (**int** i = 0; i < size; ++i) {

**int** min = list[i];

**int** index = i;

**for** (**int** j = i; j < size; ++j) {

**if** (list[j] < min) {

min = list[j];

index = j;

}

}

list[index] = list[i];

list[i] = min;

}

}

**int** \*cpyArr(**const int** \*list, **int** size) {

**int** \*res = **new int**[size];

**for** (**int** i = 0; i < size; ++i) {

res[i] = list[i];

}

**return** res;

}

**double** median(**const int** \*list, **int** size) {

**int** \*temp = cpyArr(list, size);

selSort(temp, size);

**double** res = size % 2 == 0 ? (temp[size / 2] + temp[size / 2 - 1]) / 2.0 : temp[size / 2];

**delete**[] temp;

**return** res;

}

**int** mode(**const int** \*list, **int** size) {

**if** (size <= 1) **return** -1;

**int** \*temp = cpyArr(list, size);

**int** maxOcc = temp[0];

**int** maxFrq = 1;

**int** z = 1;

**int** prev = temp[0];

**for** (**int** i = 1; i < size; ++i) {

**if** (temp[i] == prev) {

z++;

**if** (z >= maxFrq) {

maxFrq = z;

maxOcc = temp[i];

}

} **else** z = 1;

}

**int** res = maxFrq > 1 ? maxOcc : -1;

**delete**[] temp;

**return** res;

}

**int** main() {

**int** STUDENTS = 0;

cout << "How many STUDENTS are surveyed?:\t";

**do** {

cin >> STUDENTS;

**if** (STUDENTS < 0)

cout << "Invalid number of STUDENTS. Try again.";

} **while** (STUDENTS < 0);

**int** \*data = **new int**[STUDENTS];

**for** (**int** i = 0; i < STUDENTS; ++i) {

cout << "How many movies student " << i + 1

<< " sees in a month?\t";

cin >> data[i];

}

cout << "\n\n===============================\n";

// Average

**double** sum = 0;

**for** (**int** j = 0; j < STUDENTS; ++j) {

sum += data[j];

}

cout << "Students see an average of "

<< sum / STUDENTS << " movies a month.";

// Median

cout << "Students see a median of "

<< median(data, STUDENTS) << " movies a month.";

// Mode

cout << "Students see a mode of "

<< mode(data, STUDENTS) << " movies a month.";

**delete**[] data;

**return** 0;

}

## Other

#### 1. Write a function arrCmp using two pointers as arguments to compare two arrays of the same size. It returns true if all the elements at the same subscript in two array are identical and false if not.

#### Demot it in your program, for example - arrCmp(a,b,SIZE) → true

**#include <iostream>**

**using namespace** std;

**bool** arrCmp(**const int** \*a, **const int** \*b, **int** size) {

**for** (**int** i = 0; i < size; i++)

**if** (a[i] != b[i]) **return false**;

**return true**;

}

**int** main() {

**int** size;

cout << "Enter the size of 2 arrays:\t";

**do** {

cin >> size;

**if** (size <= 0)

cout << "Invalid size! Try again.";

} **while** (size <= 0);

**int** \*i1 = **new int**[size], \*i2 = **new int**[size];

**for** (**int** i = 0; i < size; ++i) {

cout << "Enter integer element " << i << " of array 1:\t";

cin >> i1[i];

}

cout << "\n\n";

**for** (**int** i = 0; i < size; ++i) {

cout << "Enter integer element " << i << " of array 2:\t";

cin >> i2[i];

}

cout << "\n\n2 arrays are " << (arrCmp(i1, i2, size) ? "similar." : "different.");

**delete**[] i1;

**delete**[] i2;

**return** 0;

}

#### 2. Write a program that use a pointer for finding the maximum and minimum element of an array. Then it exchange them and display the result to screen.

**#include <iostream>**

**using namespace** std;

**void** maxminSwap(**int** \*list, **int** size) {

**int** max = list[0], min = list[0];

**int** idMax = 0, idMin = 0;

**for** (**int** i = 1; i < size; ++i) {

**if** (list[i] > max) {

max = list[i];

idMax = i;

}

**if** (list[i] < min) {

min = list[i];

idMin = i;

}

}

**int** temp = list[idMax];

list[idMax] = list[idMin];

list[idMin] = temp;

}

**int** main() {

**const int** SIZE = 7;

**int** list[SIZE] = {2, 5, 7, 1, 3, 9, 1};

cout << "Giving an array comprises of:\t";

**for** (**int** i = 0; i < SIZE; ++i) {

cout << list[i] << ", ";

}

maxminSwap(list, SIZE);

cout << "\nAfter swap max and min, it becomes:\t";

**for** (**int** i = 0; i < SIZE; ++i) {

cout << list[i] << ", ";

}

**return** 0;

}

#### 3. Write a function (not a main), with the following header line:

##### void add\_a\_minute(int \*h, int \*m, char \*ap)

##### This function is passed two addresses of integer variables (containing an hour [h:1-12] and a minute [m:0-59]) and the addresses of a character variable [ap: either ‘a’ for AM or ‘p’ for PM]. The data pointed to by the parameters represents the time on a 12-hour clock.

##### The function advances the time by one minute.

###### If this should cause the minute to be 60, then the hour should advanced by one and the minute should be reset to 0.

If, in turn, this should cause the hour to be 13, then the hour should be reset to 1.

If, in the course of changing the hour, the hour changes from 11 to 12, then the AM/PM indicator should be changed from ‘a’ to ‘p’, or vice versa, depending on its current setting.

For example, if we had the following variables:

int hour = 11, minute = 59;

char am\_or\_pm = ‘p’;

The the statement:

add\_a\_minute(&hour, &minute, &am\_or\_pm);

Would change hour to 12, minute to 0, and am\_or\_pm to ‘a’.

Similarly, if we reset the variables:

hour = 12;

minute = 59;

am\_or\_pm = ‘a’;

Then the statement:

add\_a\_minute(&hour, &minute, &am\_or\_pm);

Would change hour to 1, change minute to 0 and leave am\_or\_pm at ‘a’.

A second call:

add\_a\_minute(&hour, &minute, &am\_or\_pm);

Would then leave hour at 1, change minute to 1 and leave am\_or\_pm at ‘a’.

**#include <iostream>**

**using namespace** std;

**void** add\_a\_minute(**int** \*h, **int** \*m, **char** \*ap) {

\*ap = \*ap == 'a' || \*ap == 'p' ? \*ap : 'a';

\*m += 1;

**if** ((\*m %= 60) == 0) {

\*h += 1;

}

**if** (\*h % 12 == 0) {

\*ap = \*ap == 'a' ? 'p' : 'a';

\*h = 12;

} **else** \*h %= 12;

}

**int** main() {

**int** hour = 11, minute = 59;

**char** am\_or\_pm = 'p';

// 11:59 PM

cout << "Add 1 minute to "

<< hour << ':' << minute << ' ' << "PM";

add\_a\_minute(&hour, &minute, &am\_or\_pm);

// 12:00 AM

cout << " and it becomes "

<< hour << ':' << minute << ' '

<< (am\_or\_pm == 'a' ? "AM" : "PM") << endl;

hour = 12;

minute = 59;

am\_or\_pm = 'a';

// 12:59 AM

cout << "Then reset to "

<< hour << ':' << minute << ' ' << "AM. ";

cout << "Add 1 minute to "

<< hour << ':' << minute << ' ' << "AM";

add\_a\_minute(&hour, &minute, &am\_or\_pm);

// 01:00 AM

cout << " and it becomes "

<< hour << ':' << minute << ' '

<< (am\_or\_pm == 'a' ? "AM" : "PM") << endl;

cout << "Add 1 minute to "

<< hour << ':' << minute << ' ' << "AM";

add\_a\_minute(&hour, &minute, &am\_or\_pm);

// 01:01 AM

cout << " and it becomes "

<< hour << ':' << minute << ' '

<< (am\_or\_pm == 'a' ? "AM" : "PM") << endl;

**return** 0;

}