Lab 4 – Making a Dynamic Array

In this exercise, you will get practice implementing a selection sort and in creating a dynamic array. Dynamic arrays will play an important part in this course and subsequent assignments, so make sure to finish this exercise.

#### Instructions

1. The attached program uses a dynamic array that will allow you sort as many numbers as you want[[1]](#footnote-2) in order from lowest to highest.

2. While the program has the proper declarations for the swap\_values and index\_of\_smallest, there are no implementations for the functions so you will need to implement these based on your CS150 skills.

3. Use these helper routines you created in part 2 to implement a sort routine.

4. Test your program on a small data set to see that it correctly sorts the numbers.

5. Answer the following three questions

#### Questions

1. What is the computational complexity of your sort routine?

I used a selection sort which has a time complexity of O(n^2)

2. Is there a way to surmise this from the code?

Big O of n raised to number of for loops.

3. Put breakpoints before and after the deletion of the array. Use the task manager to monitor the memory usage for your program. How does it change after you delete the memory used by the array[[2]](#footnote-3)?

My task manager showed no noticeable change in memory usage. It shouldn’t matter, but the computer I tested it on was a non-Windows computer. I also tested it with arrays of size 3, 7, 20, and 50.

//Sorts a list of numbers entered at the keyboard.

#include <iostream>

#include <cstdlib>

#include <cstddef>

using namespace std;

typedef int\* IntArrayPtr;

void fill\_array(int a[], int size);

//Precondition: size is the size of the array a.

//Postcondition: a[0] through a[size-1] have been

//filled with values read from the keyboard.

void sort(int a[], int size);

//Precondition: size is the size of the array a.

//The array elements a[0] through a[size-1] have values.

//Postcondition: The values of a[0] through a[size-1] have been rearranged

//so that a[0] <= a[1] <= ... <= a[size-1].

void swap\_values(int& v1, int& v2);

//Interchanges the values of v1 and v2.

int index\_of\_smallest(const int a[], int start\_index, int number\_used);

//Precondition: 0 <= start\_index < number\_used. References array elements have

//values.

//Returns the index i such that a[i] is the smallest of the values

//a[start\_index], a[start\_index + 1], ..., a[number\_used - 1].

int main( )

{

cout << "This program sorts numbers from lowest to highest.\n";

int array\_size;

cout << "How many numbers will be sorted? ";

cin >> array\_size;

IntArrayPtr a;

a = new int[array\_size];

fill\_array(a, array\_size);

sort(a, array\_size);

cout << "In sorted order the numbers are:\n";

for (int index = 0; index < array\_size; index++)

cout << a[index] << " ";

cout << endl;

system("pause");

delete [] a;

system("pause");

return 0;

}

//Uses the library iostream:

void fill\_array(int a[], int size)

{

using namespace std;

cout << "Enter " << size << " integers.\n";

for (int index = 0; index < size; index++)

cin >> a[index];

}

1. Well, subject to memory constraints.  [↑](#footnote-ref-2)
2. Some operating systems don’t show a difference, so don’t worry. Report what you see honestly, but accurately. [↑](#footnote-ref-3)