Homework 5

Programming Language Concepts

Due April 17, 2020

Name: Harry Park

1. (points) Write a program in the language of your choice that behaves differently if the language used name equivalence than if it used structural equivalence.
2. (points) Write a simple program in C++ to investigate the safety of its enumeration types. Include at least 10 different operations on enumeration types to determine what incorrect or just silly things are legal. Now, write a C program that does the same things and run it to determine how many of the incorrect or silly things are legal. Compare your results.
3. (points) Write a C program that does a large number of references to elements of two- dimensioned arrays, using only subscripting. Write a second program that does the same operations but uses pointers and pointer arithmetic for the storage- mapping function to do the array references. Compare the time efficiency of the two programs. Which of the two programs is likely to be more reliable? Why?
4. (points) Analyze and write a comparison of using C++ pointers and Java reference variables to refer to fixed heap- dynamic variables. Use safety and convenience as the primary considerations in the comparison.

1.

```
class Department {
  public String name;
  public String address;
}

class School {
  public String name;
  public String address;
}

public class EquivalenceTester {

   public static void main(String[] args) {

      Department d1 = new Department();
      School s1;

      d1.name = "Starbucks";
      d1.address = "1912 Pike Place";
      try {
            s1 = d1;
            System.out.println(s1.name + "  " + s1.address);
         } catch(Exception e) {
            System.out.println("Language uses name
equivalence.");
         }
     }
}
```

If the language uses structural equivalence, the assignment will take place. If it uses name equivalence, it will throw an error.

Inside the main method, an object of class department is created and a reference of class school is stored. When trying to assign object of department to object of school, if the assignment takes place, it means the language uses structural equivalence. Otherwise, if an exception is thrown, then the language uses name equivalence.

2.

```
#include <iostream>
using namespace std;

enum Example {
   ONE = 1,
   TWO = 2,
   FOUR = 4,
   EIGHT = 8,
   COUNT = 16,
    VAL = 5
};

int main(void) {

   cout << (ONE + TWO) << endl;   // 3
   cout << (FOUR - TWO) << endl;  // 2
   cout << (TWO * EIGHT) << endl; // 16
   cout << (EIGHT / TWO) << endl; // 4

   cout << (ONE | TWO) << endl;   // 3
   cout << (TWO & FOUR) << endl;  // 0
   cout << (TWO ^ EIGHT) << endl; // 10
   cout << (EIGHT << 1) << endl;  // 16
   cout << (EIGHT >> 1) << endl;  // 4

   int intArray[COUNT];  // valid
   int ans = 5 + VAL;  // valid

   return 0;
}
```

```
#include <stdio.h>

enum Example {
   ONE = 1,
   TWO = 2,
   FOUR = 4,
   EIGHT = 8,
   COUNT = 16,
   VAL = 5
};

int main()
{
   printf("%d\n", (ONE + TWO));  // 3
   printf("%d\n", (FOUR - TWO));  // 2
   printf("%d\n", (TWO * EIGHT));  // 16
   printf("%d\n", (EIGHT / TWO));  // 4

   printf("%d\n", (ONE | TWO));  // 3
   printf("%d\n", (TWO & FOUR));  // 0
   printf("%d\n", (TWO ^ EIGHT));  // 10
   printf("%d\n", (EIGHT << 1));  // 16
   printf("%d\n", (EIGHT >> 1));  // 4

   int intArray[COUNT];  // valid
   int ans = 5 + VAL;  // valid

   return 0;
}
```

3.

```
#include <stdio.h>
#include <time.h>
int main() {

    clock_t start, end;
    double time;

    start = clock();
    for (int i = 0; i < 10000; i++) {
        subscript();
    }
    end = clock();
    time = ((double) (end - start)) / CLOCKS_PER_SEC;          // 0.000001 s
    printf("Time taken with subscripting: %f\n", time);

    start = clock();
    for (int I = 0; I < 10000; I++) {
        pointers();
    }
    end = clock();
    time = ((double) (end - start)) / CLOCKS_PER_SEC;          // 0.000000 s
    printf("Time taken with pointers: %f\n", time);

    return 0;
}

void subscript() {
    int arr[15][5] = {0};
    for (int j = 0; j < 5; j++) {
        for (int i = 0; i < 15; i++) {
            arr[i][j];
        }
    }
    return;
}

void pointers() {
    int arr[15][5] = {0};
    for (int j = 0; j < 5; j++) {
        for (int i = 0; i < 15; i++) {
            *(*(arr + i) + j);
        }
    }
    return;
}
```

Using pointers is more time-efficient.
Using pointers would be more reliable based on the following reasons:

First, it allows management of structures which are allocated memory dynamically.
Second, it allows passing of arrays and strings to functions more efficiently.
Finally, it makes possible to pass address of structure instead of entire structure to the functions.

4.

In C and C++, pointers can be used the same way as the address.
This design offers no solution to the problem pointers dangling or lost heap-dynamic variable.
Reference type, such as Java and C# provide heap management pointers without any problems.