**Homework 5 – CS60 Linnell**

**Arjun Kohli**

**W1579330**

**Problem 1:**

**Class File (dbiguint.cpp):**

void dbiguint::operator \*=(const dbiguint & b) {

int m;

dbiguint temp;

dbiguint temp2;

int counter;

temp.reserve(19);

for (size\_t i = 0; i < capacity\_; ++i) {

for (size\_t j = 0; j < temp.size(); ++j) {

temp.data\_[j] = 0;

}

for (size\_t k = 0; k < b.size(); ++k) {

m = (data\_[i] \* b.data\_[k]);

temp.data\_[i+k] += (m % 10);

temp.data\_[i+k+1] += (m / 10);

}

temp2 += temp;

}

counter = 0;

for (size\_t l = temp2.capacity\_-1; l > -1; --l) {

if (temp2.data\_[l] == 0) {

counter++;

} else {

break;

}

}

temp2.reserve(temp2.size()-counter);

\*this = temp2;

}

dbiguint operator \*(const dbiguint &b1, const dbiguint &b2) {

int m;

int temp[20];

dbiguint temp2;

int counter;

temp2.reserve(19);

std::string s = "";

for (size\_t i = 0; i < b1.size(); ++i) {

for (size\_t j = 0; j < 19; ++j) {

temp[j] = 0;

}

for (size\_t k = 0; k < b2.size(); ++k) {

m = (b1[i] \* b2[k]);

temp[i+k] += (m % 10);

temp[i+k+1] += (m / 10);

}

s = toString(temp);

dbiguint temp3(s);

temp2 += temp3;

}

counter = 0;

for (size\_t l = temp2.size()-1; l > -1; --l) {

if (temp[l] == 0) {

counter += 1;

} else {

break;

}

}

temp2.reserve(temp2.size()-counter);

return temp2;

}

**Main File (main.cpp):**

#include "dbiguint.h"

#include <iostream>

using namespace std;

int main() {

dbiguint b1("5");

dbiguint b2("100");

dbiguint b3 = (b1\*b2);

cout << b3 << endl;

b1\*=b2;

cout << b1 << endl;

//10500

//10500

return 0;

}

**Problem 2:**

**Header File (duset.h):**

#ifndef DUSET\_H

#define DUSET\_H

#include "dynamicbag.h"

using namespace std;

class dUSet: public DynamicBag {

public:

dUSet();

dUSet(DynamicBag b);

void insert(int target);

private:

};

#endif

**Class File (duset.cpp):**

#include "dynamicbag.h"

#include "duset.h"

dUSet::dUSet(): DynamicBag() {

}

dUSet::dUSet(DynamicBag b): DynamicBag(b) {

DynamicBag temp(b);

for (size\_t i = 0; i < temp.size(); ++i) {

for (size\_t j = 0; j < temp.size(); ++j) {

if (i != j) {

if (temp[i] == temp[j]) {

temp.erase\_one(b[j]);

}

}

}

}

\*this = temp;

}

void dUSet::insert(int target) {

DynamicBag temp(\*this);

bool replica = false;

for (size\_t i = 0; i < temp.size(); ++i) {

if (temp[i] == target) {

replica = true;

}

}

if (replica == false) {

temp.insert(target);

}

\*this = temp;

}

**Main File (main.cpp):**

#include "duset.h"

#include <iostream>

#include <cstdlib>

using namespace std;

int main(){

DynamicBag b;

b.insert(5);

b.insert(70);

for(int i=0; i<5; i++) {

b.insert(i+1);

}

cout<<b<<endl;

b.erase(5);

cout<<b<<endl;

}

**Output:**

5 70 1 2 3 4

70 1 2 3 4

**Problem 3:**

**Header File (dset.h):**

#ifndef DSET\_H

#define DSET\_H

#include "duset.h"

class dSet: public dUSet {

public:

dSet();

dSet(const DynamicBag &b);

};

**Class File (dset.cpp):**

#include "duset.h"

#include "dset.h"

dSet::dSet(): dUSet() {

}

dSet::dSet(DynamicBag b): dUSet() {

DynamicBag temp;

int min = b[0];

size\_t j = 0;

for (size\_t i = 0; i < b.size(); ++i) {

if (b[i] < max) {

min = b[i];

temp[j] = min;

b.erase\_one(min);

++j;

}

}

\*this = temp;

}

void dUSet::insert(int target) {

DynamicBag temp(\*this);

bool replica = false;

for (size\_t i = 0; i < temp.size(); ++i) {

if (temp[i] == target) {

replica = true;

}

}

if (replica == false) {

if (temp[i-1] < target && temp[i+1] > target) {

temp.insertAt(target, i);

}

}

\*this = temp;

}

**Main File (main.cpp):**

#include "dset.h"

#include <iostream>

#include <cstdlib>

using namespace std;

int main(){

DynamicBag b;

b.insert(5);

b.insert(70);

for(int i=0; i<5; i++) {

b.insert(i+1);

}

cout<<b<<endl;

b.erase(5);

cout<<b<<endl;

}

**Output:**

1 2 3 4 5 70

1 2 3 4 70