**C++ Part II (INFO1-CE9265) Spring 2015 – Homework 6**

Clement Chan

**Question 4:**

#include <iostream>

#include <cstring>

#include <stdio.h>

using namespace std;

template <class T>

T calc\_abs(char num\_ops, T x, T y){

if(num\_ops == '+'){

T total = x + y;

if(total <0){

total = -(total);

}

throw total;

}

else if(num\_ops == '-'){

T total = x - y;

if(total <0){

total = -(total);

}

throw total;

}

else if(num\_ops == '\*'){

T total = x \* y;

if(total <0){

total = -(total);

}

throw total;

}

else if(num\_ops == '/'){

if(y == 0){

throw "No";

}

else{

T total = x / y;

if(total < 0){

total = -(total);

}

throw total;

}

}

}

template <class T>

void abs\_tc\_final(char num\_ops, T x, T y){

try{

calc\_abs(num\_ops, x, y);

}

catch(T a){

cout<<"The total absolute function is: " << a << endl;

}catch(...){

cout<<"Numbers cannot divide by zero.." << endl;

}

}

int main(){

cout<<"Results for float variables: " <<endl;

float x = 10.32;

float y = 5.21;

abs\_tc\_final('+', x, y);

abs\_tc\_final('-', x, y);

abs\_tc\_final('\*', x, y);

abs\_tc\_final('/', x, y);

cout<<" " <<endl;

cout<<"Results for int variables: " <<endl;

int x1 = 10;

int y1 = 3;

abs\_tc\_final('+', x1, y1);

abs\_tc\_final('-', x1, y1);

abs\_tc\_final('\*', x1, y1);

abs\_tc\_final('/', x1, y1);

cout<<" " <<endl;

cout<<"Results for double variables: " <<endl;

double x2 = 5.567;

double y2 = 6.689;

abs\_tc\_final('+', x2, y2);

abs\_tc\_final('-', x2, y2);

abs\_tc\_final('\*', x2, y2);

abs\_tc\_final('/', x2, y2);

cout<<" "<<endl;

cout<<"Results for char variables: " <<endl;

char x3 = 'm';

char y3 = 'n';

abs\_tc\_final('+', x3, y3);

abs\_tc\_final('-', x3, y3);

abs\_tc\_final('\*', x3, y3);

abs\_tc\_final('/', x3, y3);

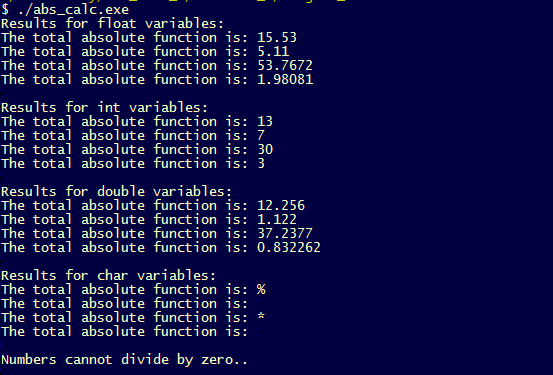
cout<<" "<<endl;

//Check for Exceptions

abs\_tc\_final('/', 5, 0);

}

Output



**Question 5:**

Item\_Class.h

#ifndef ITEM\_SET\_H

#define ITEM\_SET\_H

#include <iostream>

#include <cmath>

#include <stdlib.h>

#include <cstring>

using namespace std;

template <class T>

class set\_item{

private:

T \*items;

int size;

public:

set\_item();

set\_item(int the\_size);

set\_item(const set\_item &st);

~set\_item();

//Setters and getters

void add\_new\_item(T the\_item);

int get\_num\_items() const;

void show\_items() const;

};

#endif

Item\_Class.cpp

#include "item\_class.h"

template <class T>

set\_item <T> :: set\_item(){

cout << "This is Default Constructor ... " << endl;

items = new T[10];

size = 0;

}

template <class T>

set\_item <T> :: set\_item(int the\_size){

items = new T[the\_size];

size = the\_size;

}

template <class T>

set\_item <T> :: set\_item(const set\_item &s\_t){

size = s\_t.get\_num\_items();

delete[] items;

items = new T[size];

}

template <class T>

set\_item <T> :: ~set\_item(){

cout << "This is Destructor ... " << endl;

delete[] items;

}

template <class T>

void set\_item <T> :: add\_new\_item(T the\_item){

items[size] = the\_item;

size ++;

}

template <class T>

int set\_item <T> :: get\_num\_items() const{

return size;

}

template <class T>

void set\_item <T> :: show\_items() const{

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

cout << "The entered items in the set are: " << items[i] << endl;

}

}

Main.cpp

#include "item\_class.h"

#include "item\_class.cpp"

int main(){

set\_item<int> ST;

set\_item<double> ST\_double;

set\_item<string> ST\_str;

set\_item<char> ST\_char;

cout<< " " <<endl;

/\*\*\*\*\*\*\*\*\* For Integer \*\*\*\*\*\*\*\*\*\*\*\*/

cout<< "For Integer ... " <<endl;

ST.add\_new\_item(3);

ST.add\_new\_item(5);

ST.add\_new\_item(6);

ST.add\_new\_item(15);

ST.add\_new\_item(150);

cout<<"The number of items entered: "<<ST.get\_num\_items()<<endl;

ST.show\_items();

cout<< " " <<endl;

/\*\*\*\*\*\*\*\*\*\* For double \*\*\*\*\*\*\*\*\*\*\*\*/

cout<< "For double ... " <<endl;

ST\_double.add\_new\_item(4.5);

ST\_double.add\_new\_item(5.111);

ST\_double.add\_new\_item(2.332);

ST\_double.add\_new\_item(1.234);

ST\_double.add\_new\_item(0.5318);

cout<<"The number of items entered: " << ST\_double.get\_num\_items()<<endl;

ST\_double.show\_items();

cout<< " " <<endl;

/\*\*\*\*\*\*\*\* For Strings \*\*\*\*\*\*\*\*\*\*/

cout<< "For Strings ... " <<endl;

string a = "hi, how are you?";

string b = "ok, I am fine";

string c = "Today is freezing";

string d = "So hot!";

ST\_str.add\_new\_item(a);

ST\_str.add\_new\_item(b);

ST\_str.add\_new\_item(c);

ST\_str.add\_new\_item(d);

cout<<"The number of items entered: " << ST\_str.get\_num\_items()<<endl;

ST\_str.show\_items();

cout<< " " <<endl;

/\*\*\*\*\*\*\* For char \*\*\*\*\*\*\*\*\*\*\*\*/

cout<< "For char ... " <<endl;

ST\_char.add\_new\_item('a');

ST\_char.add\_new\_item('b');

ST\_char.add\_new\_item('c');

ST\_char.add\_new\_item('d');

ST\_char.add\_new\_item('e');

cout<<"The number of items entered: " << ST\_char.get\_num\_items()<<endl;

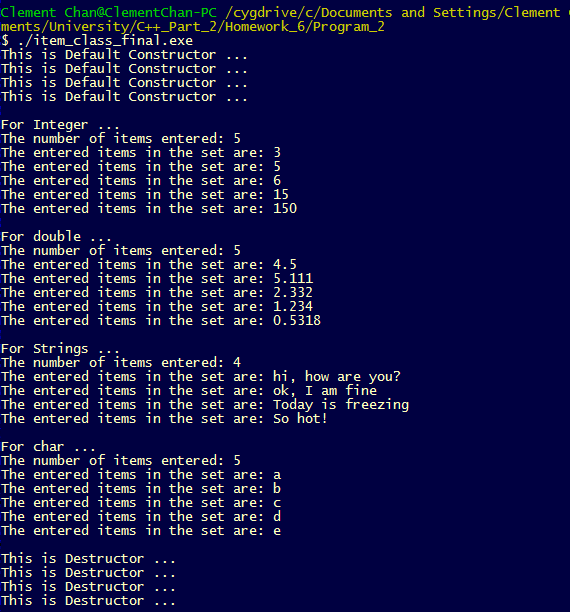
ST\_char.show\_items();

cout<< " " <<endl;

return 0;

}

Output



**Additional Question:**

template\_array.h

#ifndef Array\_H

#define Array\_H

#include <iostream>

#include <typeinfo>

template <typename T> class Array{

private:

int size;

T \*arr;

public:

Array(int the\_size); //Parameterized Constructor

void setArray(int loc, T value);

void getArray();

};

//Method Definition

template<class T>

Array<T>::Array(int the\_size){

size = the\_size;

arr = new T [size];

}

template<class T>

void Array<T> :: setArray(int loc, T value){

arr[loc] = value;

}

template<class T>

void Array<T> :: getArray(){

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

std::cout<<arr[i]<<" " << "Type: " << typeid(arr[i]).name()<<std::endl;

}

std::cout<<"------------------"<<std::endl;

}

#endif

template\_array.cpp

#include "template\_array.h"

int main(){

Array<int> Array\_int(3);

Array<float> Array\_float(5);

Array<char> Array\_char(2);

//Inputting Array for int type

Array\_int.setArray(0,3);

Array\_int.setArray(1,2);

Array\_int.setArray(2,1);

//Inputting Array for float type

Array\_float.setArray(0,0.5);

Array\_float.setArray(1,1.4);

Array\_float.setArray(2,3.23);

Array\_float.setArray(3,1.22);

Array\_float.setArray(4,4.7);

//Inputting Array for char type

Array\_char.setArray(0,'T');

Array\_char.setArray(1,'F');

//Getting different entries with different typedef

Array\_int.getArray();

Array\_float.getArray();

Array\_char.getArray();

return 0;

}

Output

