|  |
| --- |
|  |

/\* LAB #1 Recursion

\* Group: Amy Jiang

\* March 16, 2017

\* CS/IS 211

\*/

#include <iostream>

#include <string>

using namespace std;

double recursivePower(double a, int n) ;

int main()

{

cout << recursivePower(-5.5, 3);

}

/\*pre: double a is a real number

\*\* int n is a non-negative integer ( n >= 0 )

\*\*post: a double value will be returned computed by a raised to the power of n

\*/

double recursivePower(double a, int n) {

//a is the value being multipled by values of n; n cannot be neg

//base case let n = 0 when this occurs we end the call

if(n == 0) //a number raised to zero is always 1

return 1;

else {

return a\*recursivePower(a, n-1);

}

}

|  |
| --- |
|  |

#2

/\* LAB #1 Recursion

\* Group: Amy Jiang

\* March 16, 2017

\* CS/IS 211

\*/

#include <iostream>

#include <iomanip>

using namespace std;

template <typename T>

T maxArray(const T anArray[],int first, int last);

int main() {

double arr[] = { 1.5 , 2.3 , 5.6 , 2.3};

cout << fixed << setprecision(2) << maxArray(arr, 0, 3) << endl;

return 0;

}

template <typename T>

T maxArray(const T anArray[],int first, int last) {

T maxValue;

if(first == last) //\*\*BASE CASE:\*\* return value @ index: 0

maxValue = anArray[first]; //return this value since it is the only one in the array

else { //\*\*RECURSIVE CASE \*\* compares leftArr to rightArr. maxValue is returned.

int midpoint = first + (last - first)/2;

int leftArr = maxArray(anArray, first, midpoint);

int rightArr = maxArray(anArray, midpoint+1, last);

if(leftArr > rightArr) //when leftArr's value is greater than rightArr's value

maxValue = leftArr;

else if (leftArr < rightArr) //when rightArr's value is greater than leftArr's value

maxValue = rightArr;

else //leftArr == rightArr

maxValue = leftArr; //this can be either rightArr or leftArr, it doesn't matter

}

return maxValue;

}

|  |
| --- |
|  |

#3

/\* LAB #1 Recursion

\* Group: Amy Jiang

\* March 16, 2017

\* CS/IS 211

\*/

#include <iostream>

#include <string>

using namespace std;

template <typename T, typename K>

int binarySearch(const T anArray[], int first, int last, K target) {

int index = 0;

if (first > last)

index = -1; //no matches

else {

int mid = first + (last - first)/2; //get the midpoint

if(target == anArray[mid])

index = mid;

else if (target < anArray[mid])

index = binarySearch(anArray, first, mid - 1, target);

else

index = binarySearch(anArray, mid + 1, last, target);

}

return index;

}

int main()

{

//double array[] = {1.2, 2.2, 3.2 };

//char array[] = {'a','b','c'};

string array[] = {"a", "aa", "aaaa"};

int value = binarySearch(array, 0, 2, "aaaa");

cout << value << endl;

}

|  |
| --- |
|  |

#4

/\* LAB #1 Recursion

\* Group: Amy Jiang

\* March 16, 2017

\* CS/IS 211

\*/

#include <iostream>

using namespace std;

int reverseDigits(int num);

int main() {

cout << reverseDigits(456);

return 0;

}

int reverseDigits(int num){

static int sum = 0,temp = 1; //need static variables in order to keep the prev answers

//temp will increase the values place (100th , 10th, 1th)

if (num <= 0) // since int returns a decimal when theres a fraction

sum = num; // Base case: when we divide and a value of zero/neg is returned

else {

reverseDigits(num/10); //reduce num by it's right most digit

sum = sum + (num%10)\*temp;

temp = temp \*10; //will also increase by a multiple of 10 and

//will be multipled by mod and added by sum

//cout << "The value of sum is " << sum << endl;

}

return sum;

}