/\* -----------------------------------------------

Name: J-Zach Loke

Course: CMPS-385

Semester: Spring 2020

Project: No. 5 Part 1

Purpose: Linked Lists

----------------------------------------------- \*/

#include <iostream>

class LinkedList

{

private:

// given node definition

struct node

{

int age;

node \*next;

};

node \*list;

public:

LinkedList() { list = NULL; }

void push(int x)

{

/\* name: push

input: int x

output: N/A

purpose: adds an element into the linked list stack \*/

node \*p = new(node);

p->age = x;

p->next = NULL;

if (list != NULL)

{

node \*n = list;

while (n->next != NULL)

{

n = n->next;

}

n->next = p;

}

else

{

list = p;

}

}

int pop()

{

/\* name: pop

input: N/A

output: int

purpose: removes and returns an element from the linked list stack \*/

node \*p = list;

int value = p->age;

list = p->next;

return value;

}

bool isEmpty()

{

/\* name: isEmpty

input: N/A

output: bool

purpose: determins if the linked list stack is empty \*/

return (list == NULL) ? true : false;

}

};

int main()

{

/\* name: main

input: N/A

output: N/A

purpose: main function to drive the program \*/

int Age[5] = {19, 21, 17, 22, 33}; // given array definition

// 1a. Construct a linked list to hold the data in array Age

LinkedList ageStack;

for (int x: Age)

{

ageStack.push(x);

}

// 1b. Display the linked list

LinkedList tempStack; // used later for 1c

while (!ageStack.isEmpty())

{

int val = ageStack.pop();

std::cout << val << '\t';

tempStack.push(val);

}

std::cout << std::endl;

// 1c. Compute and display the average of numbers in the new linked list

int ageTotal = 0;

int eleCount = 0;

while (!tempStack.isEmpty())

{

int val = tempStack.pop();

ageTotal += val;

eleCount++;

ageStack.push(val); // used later for 1d

}

float avg = float(ageTotal)/float(eleCount);

std::cout << "The average age is " << avg << std::endl;

// 1d. Determine how many data in the linked list is above age average

int avgCount = 0;

while (!ageStack.isEmpty())

{

int val = ageStack.pop();

if (val > avg)

{

avgCount++;

}

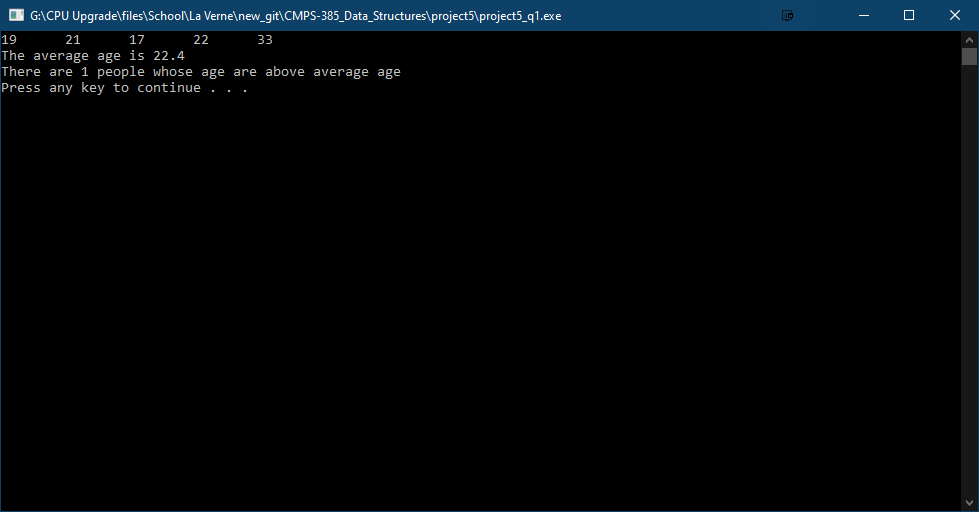
}

std::cout << "There are " << avgCount << " people whose age are above average age" << std::endl;

system("pause");

return 0;

}



/\* -----------------------------------------------

Name: J-Zach Loke

Course: CMPS-385

Semester: Spring 2020

Project: No. 5 Part 2

Purpose: Linked Lists

----------------------------------------------- \*/

#include <iostream>

#include <string>

#include <fstream>

// given node definition

struct node

{

std::string name;

int age;

node \*next;

};

class LinkedList

{

private:

node \*list;

public:

LinkedList() { list = NULL; }

void push(std::string s, int x)

{

/\* name: push

input: string s, int x

output: N/A

purpose: adds an element into the linked list stack \*/

node \*p = new(node);

p->name = s;

p->age = x;

p->next = NULL;

if (list != NULL)

{

node \*n = list;

while (n->next != NULL)

{

n = n->next;

}

n->next = p;

}

else

{

list = p;

}

}

node pop()

{

/\* name: pop

input: N/A

output: int

purpose: removes and returns an element from the linked list stack \*/

node \*p = list;

node n = \*p;

list = p->next;

return n;

}

bool isEmpty()

{

/\* name: isEmpty

input: N/A

output: bool

purpose: determines if the linked list stack is empty \*/

return (list == NULL) ? true : false;

}

};

int main()

{

/\* name: main

input: N/A

output: N/A

purpose: main function to drive the program \*/

LinkedList people;

std::fstream file;

file.open("pro5.txt", std::ios::in); // given file

while (!file.eof())

{

std::string name;

int age;

file >> name >> age;

people.push(name, age);

}

file.close();

// 2a. Display the linked list

// 2b. Display the name of the oldest person

node oldest;

oldest.age = -1;

std::cout << "Name\tAge" << std::endl;

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

while (!people.isEmpty())

{

node person = people.pop();

std::cout << person.name << '\t' << person.age << std::endl;

if (person.age > oldest.age)

{

oldest = person;

}

}

std::cout << "The oldest person is " << oldest.name << std::endl;

system("pause");

return 0;

}

