**Algorithm Analysis and Data Structures**

**CS 5343.502(Spring 2020)**

**Assignment 1**

**Divya Birla**

**2021514344**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**QUESTION:**

Make a single linked list of integers.  There should be at least 15 nodes. The list should not be sorted.

Traverse the list and then sort the list.  The list should be sorted such that your program unlinks the nodes and relinks them so that they are sorted. (DO NOT SWAP THE VALUES IN THE NODES).

Use selection sort.

Traverse the list again.

Submit the complete code.

A readme file with instructions to compile.

Screen shot of your program execution.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**SOURCE CODE:**

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Course: CS 5343.502 – Spring 2020

\* Assignment <1>

\* Name: Divya Birla

\* E-mail: dxb190021@utdallas.edu

\* Submitted: <02/8/20>

This is a d program that sorts data in a singled linked list using selection sort.

It does the sorting by delinking nodes and relinking them.

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#include<iostream>

#include<cstddef>

using namespace std;

// Definition of Nodes used in the single linked list

class Node {

public:

int element;

Node\* next;

};

// Function to insert nodes before head

Node\* insert(Node\* tmp, int ele)

{

Node\* newnode = new Node;

newnode->element = ele;

newnode->next = tmp;

return newnode;

}

// Function to delink the nodes and swap the nodes

void swapListNodes(Node \*\*refnode, Node\*head, Node\* temp1, Node\* temp2) {

(\*refnode) = temp1;

temp2->next = head;

Node \* temp3 = temp1->next;

temp1->next = head->next;

head->next = temp3;

}

// Recursive function to sort using selection sort

Node\* recselsort(Node\* refnode) {

if (refnode->next == NULL)

return refnode;

Node\* min = refnode;

Node\* prev = NULL;

Node\* temp = refnode;

while (temp->next != NULL) {

if ((temp->next->element) < (min->element)) {

min = temp->next;

prev = temp;

}

temp = temp->next;

}

if (min != refnode)

swapListNodes(&refnode, refnode, min, prev);

refnode->next = recselsort(refnode->next);

return refnode;

}

// Recrusive function to traverse and display the nodes in the linked list

void displist(Node\* lhead) {

if (lhead == NULL)

return;

else

cout << lhead->element << " ";

return displist(lhead->next);

}

int main() {

Node\* head = NULL;

int n, x;

cout << "\n--------PROGRAM TO SORT LINKED LIST BY SWAPPING NODES--------\n";

cout << "\n Enter size of the linked list(min 15)\n";

cin >> n;

// To check list size

if (n < 15)

do {

cout << "\n Min 15 lements required\n Enter a valid list size again\n" << endl;

cin >> n;

} while (n < 15);

cout << " Enter the elements of the list" << endl;

//user input for list elements

for (int i = 1; i < n; i++)

{

cout << "\n Element [" << i << "]:";

cin >> x;

head=insert(head, x);

}

cout << "\n List before sorting is as follows:\n"<<endl;

displist(head);

cout << "\n\n SORTING!!\n";

if ((head) == NULL)

{

cout << "\n List is empty\n";

}

else

head = recselsort(head);

cout << "\n List after sorting is as follows:\n";

displist(head);

cout << "\n\n\n\n";

system("pause");

return 0;

}

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**OUTPUT:**

