

|  |  |  |  |
| --- | --- | --- | --- |
| Course Code | : | SOF105 | |
| Course Name | : | Data Structure | |
| Lecturer | : | Dr. Saif Kifah | |
| Academic Session | : | 2020/09 | |
| Assessment Title | : | Assignment 1 | |
| Submission Due Date | : | 7/12/2020 | |
| Prepared by |  |  | |
| : | Student ID | Student Name |
|  | DMT2002036 | Yim Jing Xiang |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
| Date Received | : |  | |
|  |  |  | |
| Feedback from Lecturer:  Mark: | | | |

**Own Work Declaration**

I/We hereby understand my/our work would be checked for plagiarism or other misconduct, and the softcopy would be saved for future comparison(s).

I/We hereby confirm that all the references or sources of citations have been correctly listed or presented and I/we clearly understand the serious consequence caused by any intentional or unintentional misconduct.

This work is not made on any work of other students (past or present), and it has not been submitted to any other courses or institutions before.



Signature:

Date: 6/12/2020

Question 1 – **Binary Search**

Source Code

#include <iostream>

using namespace std;

int binarySearch(int \**a*, int *head*, int *tail*, int *x*)*//Function for Binary Search*

{

*//If condition is used here to prevent the mid iterator to go over the head or tail*

    if (*head* <= *tail*)

    {

*//Calculate the mid point*

        int mid = (*head* + *tail*) / 2;

*//Check if the middle value is smaller than x*

        if (*a*[mid] < *x*)

        {

*//Call recursion with head set to mid+1 and tail set to itself*

            return binarySearch(*a*, mid + 1, *tail*, *x*);

        }

*//Check if the middle value is larger than x*

        if (*a*[mid] > *x*)

        {

*//Call recursion with head set to itself and tail set to mid-1*

            return binarySearch(*a*, *head*, mid - 1, *x*);

        }

*//Check if the value is found*

        if (*a*[mid] == *x*)

        {

            return mid;

        }

    }

*//To let the result checking statement to know the value does not exist*

    return -1;

}

int main()

{

*//Declaration of variables*

    int size = 10, result, num;

    int arr[size], mid = (0 + (size - 1)) / 2;

*//Loop to accept user input into each array index*

    for (int i = size - 1; i >= 0; i--)

    {

        cout << "Please enter an integer into arr[" << i << "] : ";

        cin >> arr[i];

        cout << endl;

*/\**

*If statement -> To make sure it is in descending order*

*Else if statement -> To make sure the integer is unique*

*\*/*

        if ((arr[i] > arr[i + 1]) && i != size - 1)

        {

            cout << "The integer you have entered is larger than the previous integer.\n\n";

            i++;

        }

        else if (arr[i] == arr[i + 1] && i != size - 1)

        {

            cout << "The integer you have entered is not unique.\n\n";

            i++;

        }

    }

*//Asking user input*

    cout << "What integer are you finding in the array? ";

    cin >> num;

*//Store the returned value from the binarySearch function*

    result = binarySearch(arr, 0, size - 1, num);

*//Check result for the returned value from binarySearch*

    if (result == -1)

    {

        cout << "\nThe integer does not exist in the array.\n";

    }

    else

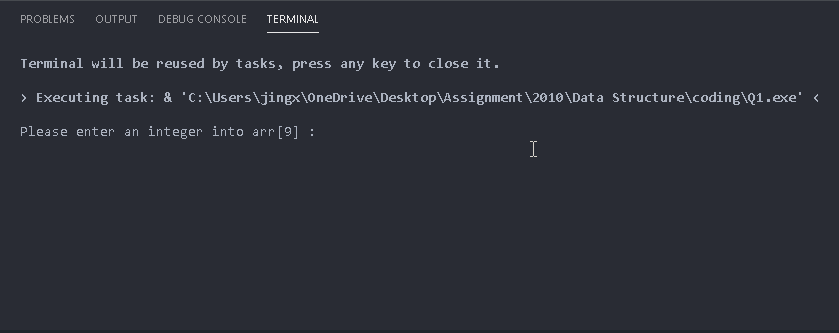
    {

        cout << "\nThe integer is located at index [" << result << "] of the array.\n\n";

    }

    return 1;

}

Result

If the user input a number larger than the previous number, the console will show that it is larger than the previous number and will prompt the user to input the number into the index of array again. If the number is already exist in the array, the console will either show that it is not an unique number or it is larger than the previous number and will prompt the user to input the number into the index of array again.

After the user has input the number that the user is finding, the program will then call the binarySearch function. If the number does exist in the array, it will return the index number of the number in the array, else it will return that the integer does not exist in the array.

Question 2 – **Recursion**

Source Code

#include <iostream>

using namespace std;

*//Function prototypes*

int fOdd(int *n*);

int fEven(int *n*);

int fOdd(int *n*)

{

*//Check if n is equals to 0 or to call the recursion*

    if (*n* == 0)

    {

        return 0;

    }

    else

    {

        return 3 + fEven(*n* - 1);

    }

}

int fEven(int *n*)

{

*//Check if n is equals to 0 or to call the recursion*

    if (*n* == 0)

    {

        return 0;

    }

    else

    {

        return 4 \* fOdd(*n* - 1);

    }

}

int main()

{

*//Declaration of variable*

    int n;

*//Ask for user input*

    cout << "Please enter the value of n: ";

    cin >> n;

    cout << endl;

*//Input validation*

    while (n < 0)

    {

        cout << "The value of n you have entered is not valid.\n";

        cout << "Please enter the value of n again (n>=0): ";

        cin >> n;

        cout << endl;

    }

*//To check if the user input is even or odd number, then trigger the function*

    if (n % 2 == 0)

    {

        cout << "The result of the mathematical formula is " << fEven(n) << ".\n\n";

    }

    else

    {

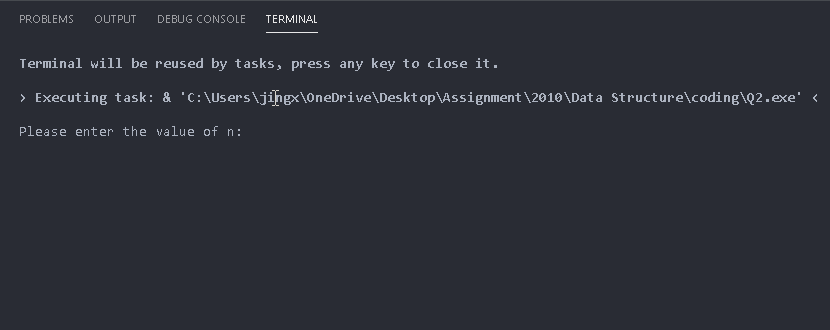
        cout << "The result of the mathematical formula is " << fOdd(n) << ".\n\n";

    }

    return 1;

}

RESULT



When the user input the value of n that is lesser than 0, the console will show that it is invalid and will prompt the user to input the value again.

Then the program will call the fEven function if the number is an even number, call the fOdd function if the number is an odd number. The program will then call the recursion again in either fEven function or fOdd function, which is mutual recursion.

Question 3 – **Merging**

Source Code

#include <iostream>

using namespace std;

*//Merge function using merging algorithm from mergesort*

void merge(int \**a*, int *size\_a*, int \**b*, int *size\_b*, int \**temp*)

{

    int i = 0, j = 0, x = 0;

*//Check if one of the loop has reached the end*

    while (i < *size\_a* && j < *size\_b*)

    {

*//To push the larger number into the temp array*

        if (*a*[i] >= *b*[j])

        {

*temp*[x] = *a*[i];

            x++;

            i++;

        }

        else if (*a*[i] < *b*[j])

        {

*temp*[x] = *b*[j];

            x++;

            j++;

        }

    }

*/\**

*While loop to push the remaining values in array a to temp array after array b has reached*

*the end from the first while loop*

*\*/*

    while (i < *size\_a*)

    {

*temp*[x] = *a*[i];

        i++;

    }

*/\**

*While loop to push the remaining values in array b to temp array after array a has reached*

*the end from the first while loop*

*\*/*

    while (j < *size\_b*)

    {

*temp*[x] = *b*[j];

        j++;

    }

}

int main()

{

*//Declaration of variables*

    int size\_1, size\_2;

*//Input for array size*

    cout << "Please enter the size of Array 1: ";

    cin >> size\_1;

*//Input validation*

    while (size\_1 <= 0)

    {

        cout << "\nPlease enter a number larger than 0.\n";

        cout << "Please enter the size of Array 1: ";

        cin >> size\_1;

    }

*//Declare array1*

    int array1[size\_1];

*//User input*

    for (int i = 0; i < size\_1; i++)

    {

        cout << "Please enter the value of array1[" << i << "]: ";

        cin >> array1[i];

        cout << endl;

*//To make sure it is in descending order*

        if ((array1[i] > array1[i - 1]) && i != 0)

        {

            cout << "The integer you have entered is larger than the previous integer.\n\n";

            i--;

        }

    }

*//Input for array size*

    cout << "Please enter the size of Array 2: ";

    cin >> size\_2;

*//Input validation*

    while (size\_2 <= 0)

    {

        cout << "\nPlease enter a number larger than 0.\n";

        cout << "Please enter the size of Array 2: ";

        cin >> size\_2;

    }

*//Declare array2*

    int array2[size\_2];

*//User input*

    for (int i = 0; i < size\_2; i++)

    {

        cout << "Please enter the value of array1[" << i << "]: ";

        cin >> array2[i];

        cout << endl;

*//To make sure it is in descending order*

        if ((array2[i] > array2[i - 1]) && i != 0)

        {

            cout << "The integer you have entered is larger than the previous integer.\n\n";

            i--;

        }

    }

*//Declare temparr*

    int temparr[size\_1 + size\_2];

*//Calls the merge function*

    merge(array1, size\_1, array2, size\_2, temparr);

    cout << "The temparr = ";

*//Printing temparr*

    for (int i = 0; i < (size\_1 + size\_2); i++)

    {

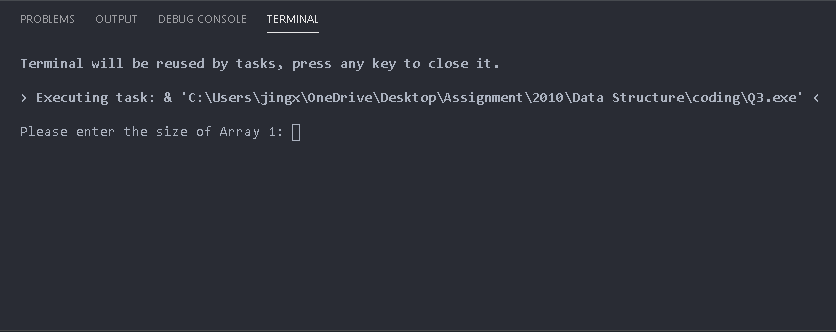
        cout << temparr[i] << " ";

    }

    cout << "\n\n";

    return 1;

}

RESULT

When the size of the array is equal or lesser than 0, the console will show that the size is invalid and will prompt the user to input the size of array again.

Everytime the user input a value into the array, the value is checked with the previous value to make sure it is in descending order.

After the user has finished input the values into both array, the program will then create a temparr and call the merge function.

After the merge function has finished pushing all the value in one of the array, it will then check and push the remaining values in one of the array into the temparr.

Question 4 – **Linked List**

Source Code

#include <iostream>

using namespace std;

*//Declare Node*

struct Node

{

    int data;

    struct Node \*next;

};

*//Declare linkedList class*

class linkedList

{

public:

*//Constructor*

    linkedList()

    {

        head->next = NULL;

        list\_size = 0;

    }

*//Public variable and functions declaration*

    int list\_size;

    void insert(int);

    void display();

    Node \*getNode(int);

    void pointNull();

private:

*//Creating a head node*

    Node \*head = new Node();

};

*//Insert function*

void linkedList::insert(int *value*)

{

*//Create a new node*

    Node \*temp1 = new Node();

    temp1->data = *value*;

    temp1->next = NULL;

*/\**

*Link the head node to the new node if the list is empty*

*Else use a traversing node to traverse to the last node*

*and link the last node to the new node*

*\*/*

    if (list\_size == 0)

    {

        head->next = temp1;

        list\_size++;

    }

    else if (list\_size > 0)

    {

        Node \*temp2 = head;

        while (temp2->next != NULL)

        {

            temp2 = temp2->next;

        }

        temp2->next = temp1;

        list\_size++;

    }

}

*//Display the values in the list*

void linkedList::display()

{

    Node \*temp = head;

    temp = temp->next;

    while (temp != NULL)

    {

        cout << temp->data << " ";

        temp = temp->next;

    }

    cout << endl;

}

*/\**

*To get the index x element in the list which*

*the head can be considered as 0-th element*

*\*/*

Node \*linkedList::getNode(int *x*)

{

    Node \*temp = head;

    int i = 1;

    while (temp != NULL)

    {

        if (i == *x*)

        {

            return temp->next;

        }

    }

    return head;

}

void linkedList::pointNull()

{

    head->next = NULL;

    list\_size = 0;

}

int main()

{

*//Declaration of variables*

    int size\_L1, size\_L2, numHold\_L1, numHold\_L2;

    linkedList L1, L2;

    Node \*temp1, \*temp2, \*tmp\_Hold1, \*tmp\_Hold2;

*//User input of L1 size*

    cout << "Input the number of list L1 nodes: ";

    cin >> size\_L1;

*//Input validation*

    while (size\_L1 <= 0)

    {

        cout << "\nPlease enter a number larger than 0.\n";

        cout << "Input the number of list L1 nodes: ";

        cin >> size\_L1;

    }

*//User input values for L1*

    cout << "Input the data elements of list L1 successively: ";

    for (int i = 0; i < size\_L1; i++)

    {

        cin >> numHold\_L1;

        L1.insert(numHold\_L1);

    }

*//Display the values inside L1*

    cout << "Nodes of List L1 are: ";

    L1.display();

*//User input of L2 size*

    cout << "\nInput the number of list L2 nodes: ";

    cin >> size\_L2;

*//Input validation*

    while (size\_L2 <= 0)

    {

        cout << "\nPlease enter a number larger than 0.\n";

        cout << "Input the number of list L2 nodes: ";

        cin >> size\_L2;

    }

*//User input values for L2*

    cout << "Input the data elements of list L2 successively: ";

    for (int i = 0; i < size\_L2; i++)

    {

        cin >> numHold\_L2;

        L2.insert(numHold\_L2);

    }

*//Display the values inside L2*

    cout << "Nodes of List L2 are: ";

    L2.display();

*/\**

*Point temp1 to the first element in L1*

*Point temp2 to the first element in L2*

*Point tmp\_Hold1 to the second element in L1*

*Point tmp\_Hold2 to the second element in L2*

*\*/*

    temp1 = L1.getNode(1);

    temp2 = L2.getNode(1);

    tmp\_Hold1 = temp1->next;

    tmp\_Hold2 = temp2->next;

*//Conditional statement to merge both L1 and L2*

    if (size\_L1 > size\_L2)

    {

        while (temp2 != NULL)

        {

            temp1->next = temp2;

            temp2->next = tmp\_Hold1;

            temp1 = tmp\_Hold1;

            tmp\_Hold1 = tmp\_Hold1->next;

            temp2 = tmp\_Hold2;

            if (tmp\_Hold2 != NULL)

            {

                tmp\_Hold2 = tmp\_Hold2->next;

            }

        }

    }

    else

    {

        while (temp1 != NULL)

        {

            temp1->next = temp2;

            temp1 = tmp\_Hold1;

            if (tmp\_Hold1 != NULL)

            {

                temp2->next = tmp\_Hold1;

                temp1 = tmp\_Hold1;

                tmp\_Hold1 = tmp\_Hold1->next;

                temp2 = tmp\_Hold2;

                tmp\_Hold2 = tmp\_Hold2->next;

            }

        }

    }

*//Point the head of L2 to NULL*

    L2.pointNull();

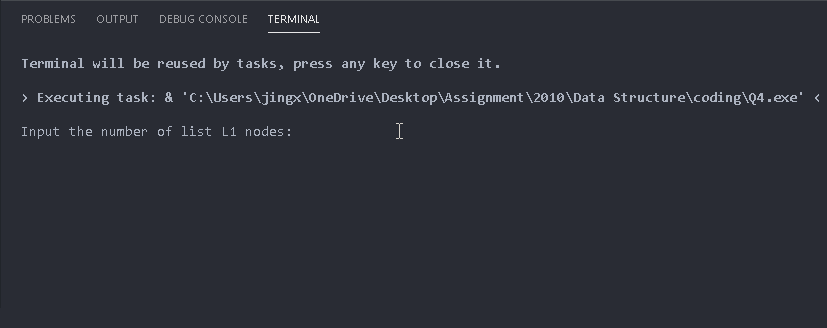
*//Display the nodes which is merged into L1*

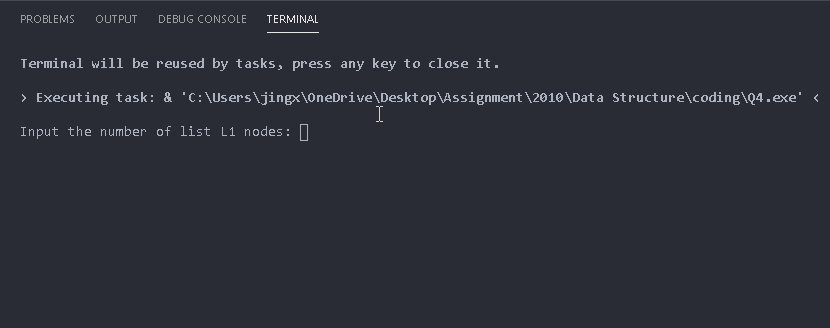
    cout << "\nThe merged nodes of L1 & L2 are: ";

    L1.display();

    return 1;

}

RESULT

In the situation where the size of L1 is larger than L2, the last element of final L1 will be the last element of L1. After that, the head node of L2 will point to NULL.

In the situation where the size of L1 is smaller or equal to L2, the last element of final L1 will be the last element of L2. After that, the head node of L2 will point to NULL.

**APPENDIX 1**

**Marking Rubrics**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Component Title** | Binary Search (Recursion) | | | | | **Percentage (%)** | **20** | |
| **Criteria** | **Score and Descriptors** | | | | | | **Weight (%)** | **Marks** |
| **(17-20)** | **(15-17)** | **(10-15)** | **(5-10)** | **(0-5)** |  |
| **C++ program and Results** | **Code free of errors.**  **Excellent use of variables and organization. Output provided** | **No errors in the coding able to compile the code. Not efficient use of variable, output not included** | **Some mistakes in the coding Not efficient use of variable, missing some tasks** | **Few mistakes in the coding Not efficient use of variable, missing few tasks** | **Many mistakes in the coding unable to compile the code, missing many tasks** |  | **20** |  |
| **TOTAL** | | | | | | | **20** |  |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Component Title** | **Recursion** | | | | | **Percentage (%)** | **20** | |
| **Criteria** | **Score and Descriptors** | | | | | | **Weight (%)** | **Marks** |
| **(17-20)** | **(15-17)** | **(10-15)** | **(5-10)** | **(0-5)** |  |
| **C++ program and Results** | **Code free of errors.**  **Excellent use of variables and organization. Output provided** | **No errors in the coding able to compile the code. Not efficient use of variable, output not included** | **Some mistakes in the coding Not efficient use of variable, missing some tasks** | **Few mistakes in the coding Not efficient use of variable, missing few tasks** | **Many mistakes in the coding unable to compile the code, missing many tasks** |  | **20** |  |
| **TOTAL** | | | | | | | **20** |  |

Note to students: Please print out and attach this appendix together with the submission of coursework

**APPENDIX 2**

**Marking Rubrics**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Component Title** | **Merging two arrays** | | | | | **Percentage (%)** | **30** | |
| **Criteria** | **Score and Descriptors** | | | | | | **Weight (%)** | **Marks** |
| **(25-30)** | **(20-25)** | **(15-20)** | **(10-15)** | **(5-10)** | **(0-5)** |
| **C++ program and Results** | **Code free of errors.**  **Excellent use of variables and organization. Output provided** | **No errors in the coding able to compile the code. Not efficient use of variable, output not included** | **Some mistakes in the coding Not efficient use of variable, missing some tasks** | **Few mistakes in the coding Not efficient use of variable, missing few tasks** | **Many mistakes in the coding unable to compile the code, missing many tasks** | **Unable to compile the program, many errors, many missing tasks** | **30** |  |
| **TOTAL** | | | | | | | **30** |  |

**Marking Rubrics**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Component Title** | **Link List** | | | | | **Percentage (%)** | **30** | |
| **Criteria** | **Score and Descriptors** | | | | | | **Weight (%)** | **Marks** |
| **(25-30)** | **(20-25)** | **(15-20)** | **(10-15)** | **(5-10)** | **(0-5)** |
| **C++ program and Results** | **Code free of errors.**  **Excellent use of variables and organization. Output provided** | **No errors in the coding able to compile the code. Not efficient use of variable, output not included** | **Some mistakes in the coding Not efficient use of variable, missing some tasks** | **Few mistakes in the coding Not efficient use of variable, missing few tasks** | **Many mistakes in the coding unable to compile the code, missing many tasks** | **Unable to compile the program, many errors, many missing tasks** | **30** |  |
| **TOTAL** | | | | | | | **30** |  |

Note to students: Please print out and attach this appendix together with the submission of coursework