CPSC 335
Assignment No.3 Part 1 Name Nicholas Ayson

1.Write a program to do the following:

a. Generate 10 random numbers between 10 and 40 and store them in linked list pointed by ALL.
b. Create an algorithm for making changes to the list ALL such that all even numbers appear at the beginning of the list. Use one of the following hints to write your program
HINT 1. While traversing ALL, collect even numbers in list EVEN and odd numbers in list ODD. Now to get the final list, simply append the two lists EVEN and ODD

Sample run:

Original list ALL: 20→37→19→33→24→38→11→13→19→24→NULL
New list ALL : 24→38→24→20→37→19→33→11→13→19→NULL

```
// 1.Write a program to do the following:
// a.   Generate 10 random numbers between 10 and 40 and store them in linked
list pointed by ALL.
// b.   Create an algorithm for making changes to the list ALL such that all even
numbers appear at the beginning of the list. Use one of the following hints to
write your program
// HINT 1. While traversing ALL, collect even numbers in list EVEN and odd
numbers in list ODD. Now to get the final list, simply append the two lists EVEN
and ODD
// Sample run:
//   Original list ALL: 20◊37◊19◊33◊24◊38◊11◊13◊19◊24◊NULL
//   New list ALL     : 24◊38◊24◊20◊37◊19◊33◊11◊13◊19◊NULL

#include <bits/stdc++.h>
using namespace std;

 //single node start
class Node
{
    public:
    int data;
    Node *next;
};

 //function to seperate even and odds
void segregateEvenOdd(Node **head_ref)
{
    Node *end = *head_ref;
    Node *prev = NULL;
    Node *curr = *head_ref;


    while (end->next != NULL)
        end = end->next;
```

```
Node *new_end = end;

while (curr->data % 2 != 0 && curr != end)
{
    new_end->next = curr;
    curr = curr->next;
    new_end->next->next = NULL;
    new_end = new_end->next;
}


if (curr->data%2 == 0)
{

    *head_ref = curr;

    while (curr != end)
    {
        if ( (curr->data) % 2 == 0 )
        {
            prev = curr;
            curr = curr->next;
        }
        else
        {

            prev->next = curr->next;


            curr->next = NULL;

            new_end->next = curr;

            new_end = curr;

            curr = prev->next;
        }
    }
}

else prev = curr;

if (new_end != end && (end->data) % 2 != 0)
{
```

```
        prev->next = end->next;
        end->next = NULL;
        new_end->next = end;
    }
    return;
}

void push(Node** head_ref, int new_data)
{
    Node* new_node = new Node();

    new_node->data = new_data;

    new_node->next = (*head_ref);

    (*head_ref) = new_node;
}

void printList(Node *node)
{
    while (node != NULL)
    {
        cout << node->data <<"->";
        node = node->next;
    }
    cout << "NULL" << endl;
}

/* Driver code*/
int main()
{
    Node* head = NULL;
    //push random number between 10 and 40
    for (int i = 0; i < 10; ++i) {
        push(&head, rand() % 40 + 10);
    }

    cout << "Original Linked list ";
    printList(head);

    segregateEvenOdd(&head);

    cout << "\nModified Linked list ";
    printList(head);
```

```
    return 0;
}
```

OUTPUT:

```
[Running] cd "c:\Users\Nick\Desktop\Algorithm-Engineering-Work\Homework 3\" &&
g++ prob1.cpp -o prob1 && "c:\Users\Nick\Desktop\Algorithm-Engineering-
Work\Homework 3\"prob1
Original Linked list 34->12->48->48->14->19->30->24->37->11->NULL

Modified Linked list 34->12->48->48->14->30->24->19->37->11->NULL

[Done] exited with code=0 in 1.459 seconds
```

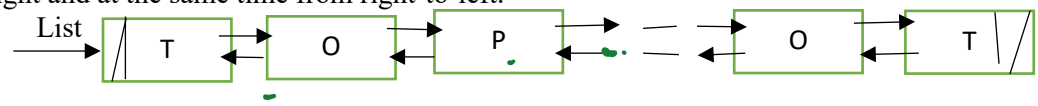
2. Use doubly linked list to determine whether a given phrase is a palindrome or not (ignore the special characters such as . , ; space , reads the same from both directions)

Sample run:

Enter a phrase: Top Spot

The phrase is a palindrome

NOTE your doubly linked list will look like this. To test for palindrome, visit the nodes from left-to-right and at the same time from right-to-left.




```
// Use doubly linked list to determine whether a given phrase is a palindrome or
not (ignore the
//            special characters such as . , ; space , reads the same from both
directions)
//       Sample run:
//       Enter a phrase: Top Spot
//            The phrase is a palindrome
//       NOTE your doubly linked list will look like this. To test for
palindrome, visit the nodes from left-to-
//       right and at the same time from right-to-left.

#include<bits/stdc++.h>
using namespace std;

// Structure of node
struct Node
{
    char data;
    struct Node *next;
    struct Node *prev;
};

/* Given a reference (pointer to pointer) to
```

```
   the head of a list and an int, inserts a
   new node on the front of the list. */
void push(struct Node** head_ref, char new_data)
{
    struct Node* new_node = new Node;
    new_node->data  = new_data;
    new_node->next = (*head_ref);
    new_node->prev = NULL;
    if ((*head_ref) !=  NULL)
      (*head_ref)->prev = new_node ;
    (*head_ref)    = new_node;
}

// Function to check if list is palindrome or not
bool isPalindrome(struct Node *left)
{
    if (left == NULL)
       return true;

    // Find rightmost node
    struct Node *right = left;
    while (right->next != NULL)
        right = right->next;

    while (left != right)
    {
        if (left->data != right->data)
            return false;

        left = left->next;
        right = right->prev;
    }

    return true;
}

// Driver program
int main()
{
    string phrase;
    struct Node* head = NULL;
    cout << "Please enter a phrase: ";
    getline(cin, phrase);
    // gets rid of spaces and any other sppecial characters
    phrase.erase(remove(phrase.begin(), phrase.end(), ' '), phrase.end());
```

```
    phrase.erase(remove(phrase.begin(), phrase.end(), '.'), phrase.end());
    phrase.erase(remove(phrase.begin(), phrase.end(), ','), phrase.end());
    //get rid of all upper cases
    for_each(phrase.begin(), phrase.end(), [](char & c){
        c = ::tolower(c);
    });

    int n = phrase.length();

    // declaring character array
    char char_array[n + 1];

    // copying the contents of the
    // string to char array
    strcpy(char_array, phrase.c_str());

    for (int i = 0; i < n; i++)
        cout << char_array[i] << endl;

    for (int i = 0; i < n; i++){
        push(&head, char_array[i]);
    }

    if (isPalindrome(head))
        printf("It is Palindrome");
    else
        printf("Not Palindrome");

    return 0;
}
```

OUTPUT:

PS C:\Users\Nick\Desktop\Algorithm-Engineering-Work\Homework 3> ./prob2
Please enter a phrase: Top Spot
t
o
p
s
p
o
t
It is Palindrome
PS C:\Users\Nick\Desktop\Algorithm-Engineering-Work\Homework 3>

3. Given two sets with at most 10 elements each, write a program to enter the elements of each set in an ordered list. Find and display both sets and their intersection set (set of their common elements)

Sample I/O

Enter the elements of setA with -1 at the end: 3 9 5 8 2 -1
Enter the elements of setB with -1 at the end: 7 2 10 3 -1
setA: 2→3→5→8→9>>NULL
setB: 2→3→7→10→NULL
setAIB: 2→3→7→NULL

```
#include <iostream>

using namespace std;

//function to sort an list
void sortList(int a[], int size)
{
    //outer for loop
    for(int i=0;i<size;i++)
    {
        //inner for loop
        for (int j=0;j<size;j++)
        {
            if(a[i]<a[j])
            {
                //swap the values
                int temp = *(a+i);
                *(a+i)=*(a+j);
                *(a+j) = temp;
            }
        }
    }
}

int main()
{
    //array declaration
    int setA[11], setB[11], setAB[11];

    //variable declaration
    int sizeA, sizeB, element, i;

    //get the setA from the user
    cout<<"Enter the elements of setA with -1 at the end: ";
    sizeA=0;
    while(true)
```

```
{
    cin>>element;
    if(element == -1)
    {
        setA[sizeA] = -1;
        break;
    }
    else
    setA[sizeA++] = element;
}

//function calling
sortList(setA, sizeA);

//get the setB from user
cout<<"Enter the elements of setB with -1 at the end: ";
sizeB=0;

while(true)
{
    cin>>element;
    if(element == -1)
    {
        setB[sizeB] = -1;
        break;
    }
    else
    setB[sizeB++] = element;
}

//function calling
sortList(setB, sizeB);
i = 0;

//display the setA
cout<<"setA: ";
while(true)
{
    if(setA[i]==-1)
    {
        cout<<"NULL";
        break;
    }
cout << setA[i++] << "->";
}
```

```
    i = 0;

    //display the setB
    cout<<endl<<"setB: ";
    while(true)
    {
        if(setB[i]==-1)
        {
            cout<<"NULL";
            break;
        }
    cout << setB[i++] << "->";
    }

    //display the intersection set
    cout<<endl<<"setA|B: ";
    for(int k=0; k<sizeA; k++)
    {
        for(int j=0; j<sizeB; j++)
        {
            if(setA[k] == setB[j]){
                cout << setB[k]<<"->";
            }
        }
    }
    cout << "NULL";
    return 0;
}
```

OUTPUT: (of most common elements)

Enter the elements of setA with -1 at the end: 2 3 5 8 9 -1
Enter the elements of setB with -1 at the end: 2 3 7 10 -1
setA: 2->3->5->8->9->NULL
setB: 2->3->7->10->NULL
setA|B: 2->3->NULL
PS C:\Users\Nick\Desktop\Algorithm-Engineering-Work\Homework 3>

CPSC 335

Assignment No. 3 Part 2 Name: Nicholas Ayson

1. (40 points)Write a program to do the following

a. Insert the data in array : string monthName[12]={“Jan”,”Feb”,……,”Dec”};
   Into a BST
b. Display the tree using ***inorder traversal***
c. Find and display the ***height of the tree***
d. Display all ***ancestors*** of “Dec”
e. Display are ***descendants*** of “DE”
f. Delete the ***leaves*** of the tree
g. Display the tree ***side way***
h. ***How many nodes*** are in the new tree (write a recursive function)

```
// a.   Insert the data in array : string monthName[12]={“Jan”,”Feb”,……,”Dec”};
// Into a BST
// b.   Display the tree using inorder traversal
// c.   Find and display the height of the tree
// d.   Display all ancestors of “Dec”
// e.   Display are descendants of “DE”
// f.   Delete the leaves of the tree
// g.   Display the tree side way
// h.   How many nodes are in the new tree (write a recursive function)

#include <iostream>
#include <algorithm>
#include <iomanip>
using namespace std;

struct node
{
    string   info;
    node *left, *right;
};

void insert(node* &p, string x)//-------------------------inserting x in BST
{
    if( p==NULL)
    {
        p = new node; p->info = x;
        p->left = NULL; p->right = NULL;
    }
    else
    {
        if (x < p->info) insert(p->left, x);
```

```
        if (x > p->info) insert(p->right, x);
    }
}

void show(node* p)//---------------------------display BST using inorder
traversal
{
    if (p != NULL)
    {
        show(p->left);
        cout << p->info << "  ";
        show(p->right);
    }
}

int height(node* p)//------------------------------------ return the tree
height
{
    if (p == NULL) return -1;  //for tree level return 0
    else return 1 + max(height(p->left), height(p->right));
}

node* find(node* curr, string v) {
        if (curr) {
            if (curr->info == v) return curr;
            if (v < curr->info) return find(curr->left, v); // search left-sub-
tree
            if (v > curr->info) return find(curr->right, v); // search right-sub-
tree
        }
        return nullptr;
    }

void findDescendants(node* p, string x)//----------------------------------
search for item X
{
    node* curr = find(p, x);
    if (!curr) {
        cout << "could not find element \'" << x << "\'";
    }
    else if (!curr->left && !curr->right) {
        cout << "\'" << x << "\' is a leaf";
    }
    else {
        insert(curr->left, x);
```

```
        insert(curr->right, x);
    }

}


bool showAncestor(node* t, string target) //--display the ancestors of node with
info target
{
    if (t == NULL) return false;
    if (t->info == target) return true;

    if (showAncestor(t->left, target) ||
        showAncestor(t->right, target) )
    {
        cout << t->info << " ";
        return true;
    }
    else
        return false;
}

node* remLeaves(node* &t)//---------------------remove existing binary tree
leaves
{
    if (!t) {
        return nullptr;
    }
    if (!t->left && !t->right) {
        std::cout << t->info << " ";
        free(t);
        return nullptr;
    }

    t->left = remLeaves(t->left);
    t->right = remLeaves(t->right);

    return t;
}

void sideWay(node* t, int s)// ------------------------------ display BST
sideway
{
    if (t != NULL)
```

```
    {
        sideWay(t->right, s += 5);
        cout << setw(s) << t->info<<endl;
        sideWay(t->left, s);
    }
}

int countNodes(node* t)// ----------------------count number of nodes recursively
{
    if (t == NULL) return 0;
    else return 1 + countNodes(t->left) + countNodes(t->right);
    //to find total of all nodes replace 1 with t->info
}

//--------------------------------main ----------------------------------------
--
int main()
{
    node *t = NULL; node *t2=NULL;
    string monthName[12] = { "Jan", "Feb", "Mar", "Apr", "May", "Jun", "Jul",
"Aug", "Sep", "Oct", "Nov", "Dec" };

    //creat tree and display
    for (int i = 0; i < 12; ++i)
    {
        insert(t, monthName[i]);
    }
    cout << "InOrder Transversal: ";
    show(t);
    cout << endl;

    //find and display tree hight
    cout << "The tree hight is " << height(t) << endl;

    //ancestor call
    cout << "Ancestors of Dec: " ;
    showAncestor(t, "Dec");
    cout << endl;

    //descendent call
    cout << "Descendents of Dec: " ;
    findDescendants(t, "Dec");
    cout << endl;

    //remove leaves
```

```
    cout << "Removing Leaves: ";
    remLeaves(t);
    cout << endl;

    //display tree side-way
    cout << "Tree in sideway direction\n";
    int s = 0;
    sideWay(t, s);

    //fid and display number of nodes in tree
    cout << "No. of nodes=" << countNodes(t) << endl;

    return 0;

}
```

OUTPUT:

```
[Running] cd "c:\Users\Nick\Desktop\Algorithm-Engineering-Work\Homework 3\" &&
g++ prob4etc.cpp -o prob4etc && "c:\Users\Nick\Desktop\Algorithm-Engineering-
Work\Homework 3\"prob4etc
InOrder Transversal: Apr  Aug  Dec  Feb  Jan  Jul  Jun  Mar  May  Nov  Oct  Sep
The tree hight is 5
Ancestors of Dec: Aug Apr Feb Jan
Descendents of Dec: 'Dec' is a leaf
Removing Leaves: Dec Jul Nov
Tree in sideway direction
                Sep
                     Oct
           May
      Mar
           Jun
  Jan
      Feb
                Aug
           Apr
No. of nodes=9
```

2. (30 points)On paper(no programming) Insert the name of months from Jan to Dec in a
   a. AVL tree
   b. B-tree of order 3
   c. B^+ tree of order 3