## CSCI 140 PA 8 Submission

## Due Date: 10/21/2021 Late (date and time):\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

## Name(s): Nero Li

Exercise 1 -- need to submit source code and I/O  
 -- check if completely done ✔️ ; otherwise, discuss issues below  
Pseudocode below:

Algorithm preOrder(root):

Create a stack with type Tree\*

Create Tree\* cur = root

While stack is not empty & cur is not NULL:

If cur is not NULL:

Print cur’s elem

Push cur into stack

Move cur to his first child

Else:

Cur move to the top of the stack

If cur has other child in the middle and that child is not visited yet:

Move cursor to that child

Else:

Pop stack

Move cur to his last child

End while

Source code below:

/\* Program: PA\_8\_exercise\_1

Author: Nero Li

Class: CSCI 220

Date: 10/21/2021

Description:

Provide pseudocode for either preorder traversal or post-order traversal for

general trees without recursion.

I certify that the code below is my own work.

Exception(s): N/A

\*/

#include <iostream>

#include <stack>

using namespace std;

const int SIZE{3};

struct Tree

{

char elem;

Tree \*parent;

Tree \*child[SIZE];

bool visited;

}; // Assume maximum child amount is 3

bool noChild(Tree \*cur)

{

for (int i = 0; i < SIZE; ++i)

{

if (cur->child[i])

{

return false;

}

}

return true;

}

void preOrder(Tree \*root)

{

Tree \*cur = root;

stack<Tree\*> stk;

while (!stk.empty() || cur)

{

if (cur)

{

cur->visited = true;

cout << cur->elem << ' ';

stk.push(cur);

cur = cur->child[0];

}

else

{

cur = stk.top();

if (cur->child[1] && !cur->child[1]->visited)

{

cur = cur->child[1];

}

else

{

stk.pop();

cur = cur->child[2];

}

}

}

cout << endl;

}

void createNewNode(Tree \*p, char elem, Tree \*parent, Tree \*leftChild, Tree \*midChild, Tree \*rightChild)

{

p->elem = elem;

p->parent = parent;

p->child[0] = leftChild;

p->child[1] = midChild;

p->child[2] = rightChild;

p->visited = false;

}

int main()

{

Tree \*A = new Tree;

Tree \*B = new Tree;

Tree \*C = new Tree;

Tree \*D = new Tree;

Tree \*E = new Tree;

Tree \*F = new Tree;

createNewNode(A, 'A', NULL, B, C, D);

createNewNode(B, 'B', A, E, F, NULL);

createNewNode(C, 'C', A, NULL, NULL, NULL);

createNewNode(D, 'D', A, NULL, NULL, NULL);

createNewNode(E, 'E', B, NULL, NULL, NULL);

createNewNode(F, 'F', B, NULL, NULL, NULL);

preOrder(A);

cout << "Author: Nero Li\n";

return 0;

}

Input/output below:

A B E F C D

Author: Nero Li

Exercise 2 -- need to submit source code and I/O  
 -- check if completely done ✔️ ; otherwise, discuss issues below

Pseudocode below:

Algorithm levelOrder(root):

Create a queue with type Tree\*

Create Tree\* cur

Enqueue root node into queue

While queue is not empty:

Let \*cur equal to the first element of the queue

Dequeue

Print the element for cur

While go through all child node:

If a child node is not NULL:

Enqueue that child node

Source code below:

/\* Program: PA\_8\_exercise\_2

Author: Nero Li

Class: CSCI 220

Date: 10/21/2021

Description:

Provide pseudocode for breadth-first (level order) traversal for general trees.

I certify that the code below is my own work.

Exception(s): N/A

\*/

#include <iostream>

#include <queue>

using namespace std;

const int SIZE{3};

struct Tree

{

char elem;

Tree \*parent;

Tree \*child[SIZE];

}; // Assume maximum child amount is 3

void levelOrder(Tree \*root)

{

Tree \*cur = root;

queue<Tree\*> que;

que.push(cur);

while (!que.empty())

{

cur = que.front();

que.pop();

cout << cur->elem << ' ';

for (int i = 0; i < SIZE; ++i)

{

if (cur->child[i])

{

que.push(cur->child[i]);

}

}

}

cout << endl;

}

void createNewNode(Tree \*p, char elem, Tree \*parent, Tree \*leftChild, Tree \*midChild, Tree \*rightChild)

{

p->elem = elem;

p->parent = parent;

p->child[0] = leftChild;

p->child[1] = midChild;

p->child[2] = rightChild;

}

int main()

{

Tree \*A = new Tree;

Tree \*B = new Tree;

Tree \*C = new Tree;

Tree \*D = new Tree;

Tree \*E = new Tree;

Tree \*F = new Tree;

createNewNode(A, 'A', NULL, B, C, D);

createNewNode(B, 'B', A, E, F, NULL);

createNewNode(C, 'C', A, NULL, NULL, NULL);

createNewNode(D, 'D', A, NULL, NULL, NULL);

createNewNode(E, 'E', B, NULL, NULL, NULL);

createNewNode(F, 'F', B, NULL, NULL, NULL);

levelOrder(A);

cout << "Author: Nero Li\n";

return 0;

}

Input/output below:

A B C D E F

Author: Nero Li

Answer for Question 1:

For my first algorithm in exercise 1, the running time is O(n) where n is the total nodes that the tree has.

For my second algorithm in exercise 2, the running time is O(n) where n is the total nodes that the tree has.

Answer for Question 2:

Preorder: A B E F C D G H I

Post-order: E F B C G H I D A

Level order: A B C D E F G H I

Extra Credit

Pseudocode below:

Algorithm postOrder(root):

Create a stack with type Tree\*

Create Tree\* cur = root

Create Tree\* prev = NULL

While stack is not empty:

Let cur equal to the top of the stack

If the cur node does not have child or all of his child has been visited:

Print the element for cur

Pop stack

Let prev = cur

Else:

Push all the exist child node from cur node into stack

Source code below:

/\* Program: PA\_8\_extra\_credit

Author: Nero Li

Class: CSCI 220

Date: 10/21/2021

Description:

Provide pseudocode for either preorder traversal or post-order traversal for

general trees without recursion.

I certify that the code below is my own work.

Exception(s): N/A

\*/

#include <iostream>

#include <stack>

using namespace std;

const int SIZE{3};

struct Tree

{

char elem;

Tree \*parent;

Tree \*child[SIZE];

}; // Assume maximum child amount is 3

bool noChild(Tree \*cur)

{

for (int i = 0; i < SIZE; ++i)

{

if (cur->child[i])

{

return false;

}

}

return true;

}

bool visited(Tree \*cur, Tree \*prev)

{

int i{SIZE - 1};

while (!cur->child[i] && i >= 0)

{

--i;

}

if (prev && cur->child[i] == prev)

{

return true;

}

return false;

}

void postOrder(Tree \*root)

{

Tree \*cur = root;

Tree \*prev = NULL;

stack<Tree\*> stk;

stk.push(cur);

while (!stk.empty())

{

cur = stk.top();

if (noChild(cur) || visited(cur, prev))

{

cout << cur->elem << ' ';

stk.pop();

prev = cur;

}

else

{

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

{

if (cur->child[i])

{

stk.push(cur->child[i]);

}

}

}

}

cout << endl;

}

void createNewNode(Tree \*p, char elem, Tree \*parent, Tree \*leftChild, Tree \*midChild, Tree \*rightChild)

{

p->elem = elem;

p->parent = parent;

p->child[0] = leftChild;

p->child[1] = midChild;

p->child[2] = rightChild;

}

int main()

{

Tree \*A = new Tree;

Tree \*B = new Tree;

Tree \*C = new Tree;

Tree \*D = new Tree;

Tree \*E = new Tree;

Tree \*F = new Tree;

Tree \*G = new Tree;

Tree \*H = new Tree;

Tree \*I = new Tree;

createNewNode(A, 'A', NULL, B, C, D);

createNewNode(B, 'B', A, E, F, NULL);

createNewNode(C, 'C', A, NULL, NULL, NULL);

createNewNode(D, 'D', A, NULL, NULL, NULL);

createNewNode(E, 'E', B, NULL, NULL, NULL);

createNewNode(F, 'F', B, NULL, NULL, NULL);

postOrder(A);

cout << "Author: Nero Li\n";

return 0;

}

Input/output below:

E F B C D A

Author: Nero Li