Date: 16/03/2021

# Practical 7:

## 2CSDE75 - Advanced Data Structures

Name: Shrey Viradiya

Roll No: 18BCE259

Aim:

Implement heap data structure using linked list structure. The list should retrieve high priority object every time the extract operation is performed.

## Code:

### Prac7_MaxHeap.cpp

```cpp
// Implement heap data structure using linked list structure.
// The list should retrieve high priority object every time the extract operation is perfor
med.

#include "MaxHeap.h"
#include <iostream>

int main(){
    using namespace std;

    MaxHeap data("Ajino Motado");

    data.AddData("data.csv", 1);
    data.PrettyPrinting();

    for (int i = 0; i < 5; i++)
    {
        pair<int, int> k = data.ExtractMax();
        cout << k.first << " " << k.second << endl;
    }
    data.PrettyPrinting();

    for (int i = 0; i < 10; i++)
    {
        data.insert(i, i*10);
    }
    data.PrettyPrinting();

    return 0;
}
```

### MaxHeap.h

```cpp
#pragma once
#include <iostream>
#include <cstring>
#include <string>
#include <fstream>
#include <stdexcept> // std::runtime_error
#include <sstream>

class HeapNode
{
public:
```

```
    HeapNode *right, *left;
    int key;
    int object;

    HeapNode()
    {
        key = 0;
        object = 0;
        right = nullptr;
        left = nullptr;
    }
    HeapNode(int k, int o)
    {
        key = k;
        object = o;
        right = nullptr;
        left = nullptr;
    }

    ~HeapNode(){
        delete left;
        delete right;
    }
};

class MaxHeap{
    private:
        char name[50];
        HeapNode *root;

    public:
        MaxHeap(const char nameinput[50]);
        ~MaxHeap();
        void insert(int key, int object);
        void AddData(std::string filename, int isHeading);
        void PrettyPrinting();
        std::pair<int, int> ExtractMax();
};

MaxHeap::MaxHeap(const char nameinput[50]){
    strcpy(name, nameinput);
    root = nullptr;
}

void deleteNode(HeapNode *node){
    if (node != nullptr)
    {
        deleteNode(node->left);
```

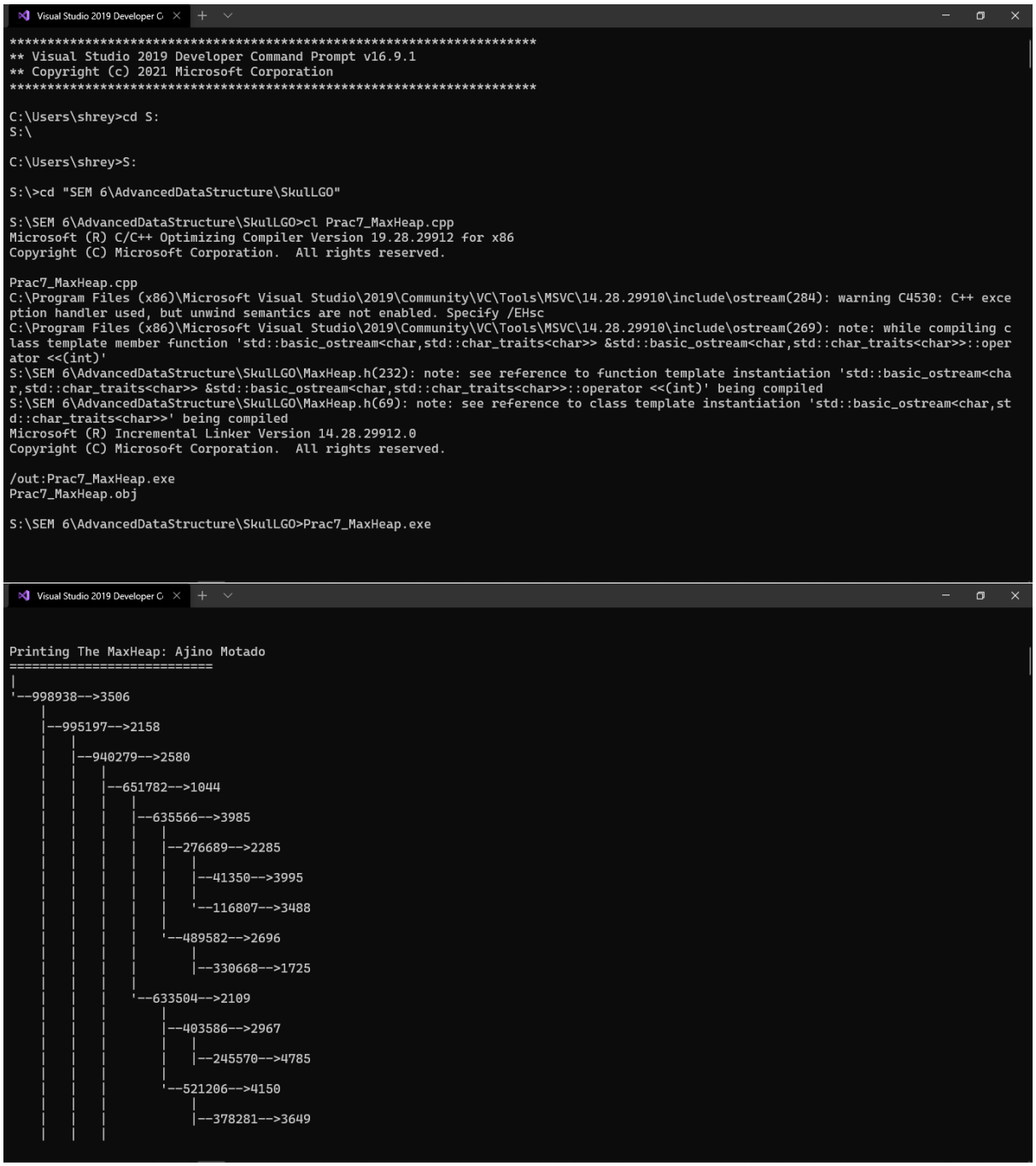
```
        deleteNode(node->right);
        delete node;
    }
}

MaxHeap::~MaxHeap()
{
    using namespace std;
    deleteNode(root);
    cout << "Memory Released of " << name << endl;
}

int countNode(HeapNode *node){
    if (node == nullptr)
    {
        return 0;
    }
    else{
        return countNode(node->left) + countNode(node->right) + 1;
    }
}

void MaxHeap::insert(int key, int object){
    if (root == nullptr)
    {
        root = new HeapNode(key, object);
        return;
    }

    HeapNode *iter = root;

    while (iter != nullptr)
    {
        if (key > iter->key)
        {
            int tk, to;
            tk = iter->key;
            to = iter->object;
            iter->key = key;
            iter->object = object;
            key = tk;
            object = to;
        }

        if (iter->left == nullptr || iter->right == nullptr)
        {
            if (iter->left == nullptr)
            {
```

```
                iter->left = new HeapNode(key, object);
                return;
            }
            else{
                iter->right = new HeapNode(key, object);
                return;
            }
        }

        if (countNode(iter->left) >  countNode (iter->right))
        {
            iter = iter->right;
        }
        else{
            iter = iter->left;
        }
    }
}

std::pair <int, int> MaxHeap::ExtractMax(){
    if (root == nullptr) return {-1, -1};

    int key = root->key, object = root->object;

    HeapNode *iter = root;
    HeapNode *prev = nullptr;

    while (iter != nullptr)
    {
        if (iter->left != nullptr && iter->right != nullptr)
        {
            if (iter->left->key > iter->right->key)
            {
                iter->key = iter->left->key;
                iter->object = iter->left->object;
                prev = iter;
                iter = iter->left;
            }
            else
            {
                iter->key = iter->right->key;
                iter->object = iter->right->object;
                prev = iter;
                iter = iter->right;
            }
        }
        else
        {
```

```
            if (iter->left != nullptr)
            {
                iter->key = iter->left->key;
                iter->object = iter->left->object;
                prev = iter;
                iter = iter->left;
            }
            else if(iter->right != nullptr){
                iter->key = iter->right->key;
                iter->object = iter->right->object;
                prev = iter;
                iter = iter->right;
            }
            else{
                if (prev == nullptr)
                {
                    delete iter;
                    iter = nullptr;
                    root = nullptr;
                }
                else{
                    prev->left == iter ? (prev->left = nullptr) : (prev->right = nullptr);
                    delete iter;
                    iter = nullptr;
                }
            }
        }
    }
    return {key, object};
}

void MaxHeap::AddData(std::string filename, int isHeading = 1){
    using namespace std;
    // working with csv in CPP
    // https://www.gormanalysis.com/blog/reading-and-writing-csv-files-with-cpp/

    ifstream myFile(filename);
    // if(!myFile.is_open()) throw runtime_error("Could not open file");

    string line, word;
    int val;

    if (isHeading)  getline(myFile, line);

    // Read data, line by line
    while(getline(myFile, line))
    {
        // Create a stringstream of the current line
```

```
        stringstream ss(line);
        pair<int, int> data;

        // add the column data
        // of a row to a pair
        getline(ss, word, ',');
        data.first = stoi(word);

        getline(ss, word, ',');
        data.second = stoi(word);

        insert(data.first, data.second);
    }

    // Close file
    myFile.close();
}

void printBT(const std::string& prefix, const HeapNode* node, bool isLeft)
{
    if( node != nullptr )
    {
        std::cout << prefix;
        std::cout << "|" << std::endl;
        std::cout << prefix;
        std::cout << (isLeft ? "|--" : "'--" );

        // print the value of the node
        std::cout << node->key << "-->" << node->object << std::endl;

        // enter the next tree level - left and right branch
        printBT( prefix + (isLeft ? "|   " : "    ") , node->left, true);
        printBT( prefix + (isLeft ? "|   " : "    ") , node->right, false);
    }
}

void MaxHeap::PrettyPrinting(){
    using namespace std;

    cout << "\n\nPrinting The MaxHeap: " << name << endl;
    cout << "==========================" << endl;
    printBT("", root, false);
}
```

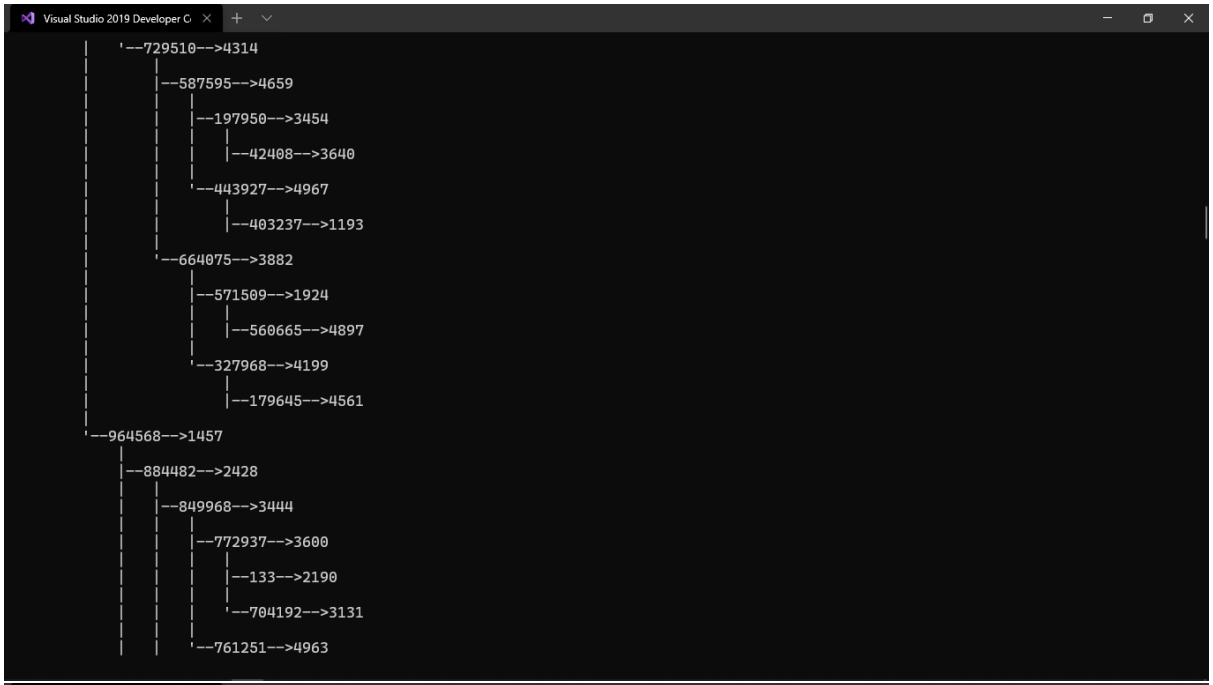
Snapshot of the output:

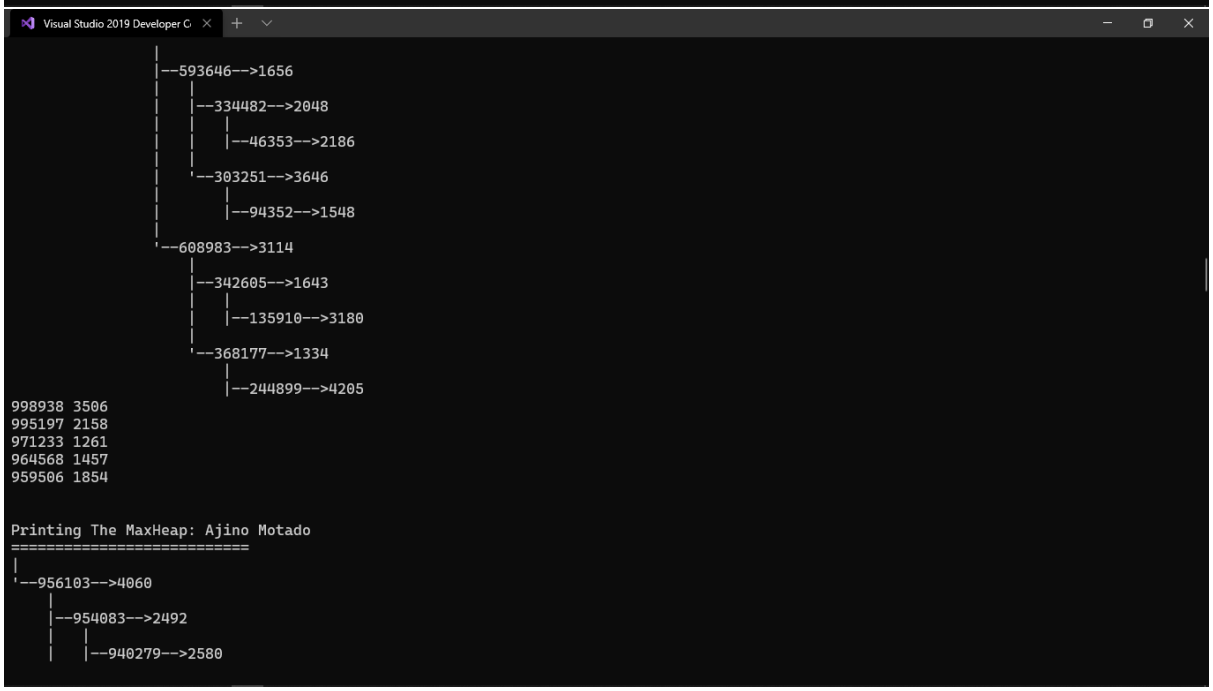


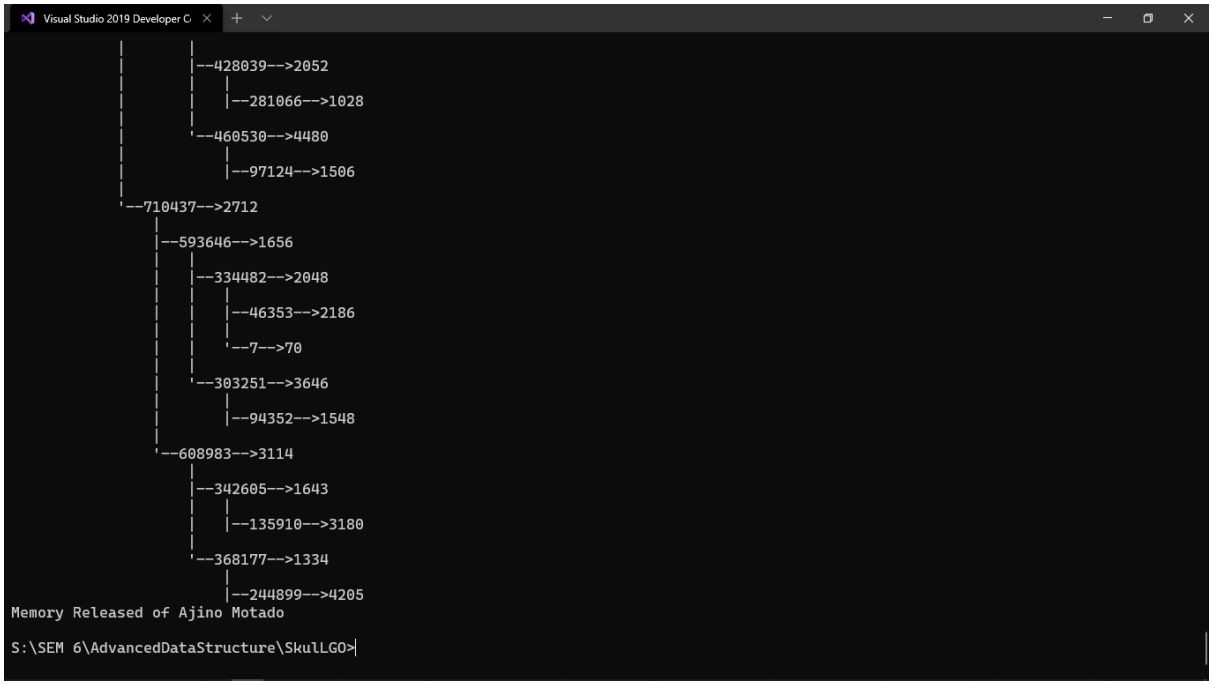
```
Visual Studio 2019 Developer C
**********************************************************************
** Visual Studio 2019 Developer Command Prompt v16.9.1
** Copyright (c) 2021 Microsoft Corporation
**********************************************************************

C:\Users\shrey>cd S:
S:\

C:\Users\shrey>S:

S:\>cd "SEM 6\AdvancedDataStructure\SkulLGO"

S:\SEM 6\AdvancedDataStructure\SkulLGO>cl Prac7_MaxHeap.cpp
Microsoft (R) C/C++ Optimizing Compiler Version 19.28.29912 for x86
Copyright (C) Microsoft Corporation.  All rights reserved.

Prac7_MaxHeap.cpp
C:\Program Files (x86)\Microsoft Visual Studio\2019\Community\VC\Tools\MSVC\14.28.29910\include\ostream(284): warning C4530: C++ exce
ption handler used, but unwind semantics are not enabled. Specify /EHsc
C:\Program Files (x86)\Microsoft Visual Studio\2019\Community\VC\Tools\MSVC\14.28.29910\include\ostream(269): note: while compiling c
lass template member function 'std::basic_ostream<char,std::char_traits<char>> &std::basic_ostream<char,std::char_traits<char>>::oper
ator <<(int)'
S:\SEM 6\AdvancedDataStructure\SkulLGO\MaxHeap.h(232): note: see reference to function template instantiation 'std::basic_ostream<cha
r,std::char_traits<char>> &std::basic_ostream<char,std::char_traits<char>>::operator <<(int)' being compiled
S:\SEM 6\AdvancedDataStructure\SkulLGO\MaxHeap.h(69): note: see reference to class template instantiation 'std::basic_ostream<char,st
d::char_traits<char>>' being compiled
Microsoft (R) Incremental Linker Version 14.28.29912.0
Copyright (C) Microsoft Corporation.  All rights reserved.

/out:Prac7_MaxHeap.exe
Prac7_MaxHeap.obj

S:\SEM 6\AdvancedDataStructure\SkulLGO>Prac7_MaxHeap.exe
```



```
Visual Studio 2019 Developer C


Printing The MaxHeap: Ajino Motado
==========================
|
'--998938-->3506
    |
    |--995197-->2158
    |   |
    |   |--940279-->2580
    |   |   |
    |   |   |--651782-->1044
    |   |   |   |
    |   |   |   |--635566-->3985
    |   |   |   |   |
    |   |   |   |   |--276689-->2285
    |   |   |   |   |   |
    |   |   |   |   |   |--41350-->3995
    |   |   |   |   |   |
    |   |   |   |   |   '--116807-->3488
    |   |   |   |   |
    |   |   |   |   '--489582-->2696
    |   |   |   |       |
    |   |   |   |       |--330668-->1725
    |   |   |   |
    |   |   |   '--633504-->2109
    |   |   |       |
    |   |   |       |--403586-->2967
    |   |   |       |   |
    |   |   |       |   |--245570-->4785
    |   |   |       |
    |   |   |       '--521206-->4150
    |   |   |           |
    |   |   |           |--378281-->3649
    |   |   |
```


Visual Studio 2019 Developer C
|   '--729510-->4314
|       |
|       |--587595-->4659
|       |   |
|       |   |--197950-->3454
|       |   |   |
|       |   |   |--42408-->3640
|       |   |
|       |   '--443927-->4967
|       |       |
|       |       |--403237-->1193
|       |
|       '--664075-->3882
|           |
|           |--571509-->1924
|           |   |
|           |   |--560665-->4897
|           |
|           '--327968-->4199
|               |
|               |--179645-->4561
|
'--964568-->1457
    |
    |--884482-->2428
    |   |
    |   |--849968-->3444
    |   |   |
    |   |   |--772937-->3600
    |   |   |   |
    |   |   |   |--133-->2190
    |   |   |   |
    |   |   |   '--704192-->3131
    |   |   |
    |   |   '--761251-->4963


Visual Studio 2019 Developer C
|
|--593646-->1656
|   |
|   |--334482-->2048
|   |   |
|   |   |--46353-->2186
|   |
|   '--303251-->3646
|       |
|       |--94352-->1548
|
'--608983-->3114
    |
    |--342605-->1643
    |   |
    |   |--135910-->3180
    |
    '--368177-->1334
        |
        |--244899-->4205
998938 3506
995197 2158
971233 1261
964568 1457
959506 1854
Printing The MaxHeap: Ajino Motado
==========
|
'--956103-->4060
    |
    |--954083-->2492
    |   |
    |   |--940279-->2580




## Conclusion:

With Heap data structure, priority queue is implemented efficiently. This data structure can be implemented with array or LinkedList.