MMU MULTIMEDIA UNIVERSITY

TCP2101 ALGORITHM DESIGN & ANALYSIS
SEMESTER 2,
YEAR 2020/2021

ASSIGNMENT 1

GROUP 3

NAME	**STUDENT ID**
CHANG KAI BOON (Team Leader)	1181101282
ANG KELVIN	1181101297
PRITESH PATEL	1181101645

Table Of Content

1.0 Data Generation

1.1 Data Generation Algorithms

Algorithm 1 randomStringWithNum

```
1: procedure randomStringWithNum(num)
2:    letter ← { all the alphabets and digits }
3:    randomString ← { }
4:    while the size of letter is not equal to num do
5:        position ← randNumber % ( letterSize -1 )
6:        letterSubset ← get subset of 1 letter on position
7:    endWhile
8:    return randomString
9:  end procedure
```

Algorithm 2 randomStringWithNoNum

```
1: procedure randomStringWithNoNum(num)
2:    letter ← { all the alphabets }
3:    randomString ← { }
4:    while the size of letter is not equal to num do
5:        position ← randNumber % ( letterSize -1 )
6:        letterSubset ← get subset of 1 letter on position
7:    endWhile
8:    return randomString
9:  end procedure
```

Algorithm 3 randomStringWithNoNum

```
1: procedure randomStringWithNoNum(num)
2:   letter ← { all the alphabets }
3:   randomString ← { }
4:   while the size of letter is not equal to num do
5:      position ← randNumber % ( letterSize -1 )
6:      letterSubset ← get subset of 1 letter on position
7:   endWhile
8:   return randomString
9: end procedure
```

Algorithm 4 randomDomain

```
1: procedure randomDomain()
2:   setDomain ← { all the domains }
3:   randomDomain ← { }
4:   position ← randNumber % ( setDomainSize )
5:   randomDomain = setDomain[position]
6:   return randomDomain
7: end procedure
```

1.2 Program of Data Generation

```
#include <iostream>
#include <thread>
#include <string>
#include <array>
#include <fstream>
using namespace std;

string randomStringWithNum(int num){

   string letter =
"ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789";
   string randomString = "";
   int pos;
   while(randomString.size() != num){
      pos = rand() % (letter.size() - 1);
      randomString += letter.substr(pos,1);
   }
   return randomString;
}

string randomStringNoNum(int num){

   string letter =
"ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz";
   string randomString = "";
   int pos;
   while(randomString.size() != num){
      pos = rand() % (letter.size() - 1);
      randomString += letter.substr(pos,1);
   }
   return randomString;
}

string randomDomain(){
   array<string, 3> setDomain {"com","my","org"};
   string randomDomain = "";
   int pos;
   pos = rand() % (setDomain.size());
   randomDomain = setDomain[pos];

   return randomDomain;
}
```

```
int main(){
   string email;
   int A = 100, B = 100000 , C=500000;


   srand(time(0));
   ofstream setA("setA.txt");
   ofstream setB("setB.txt");
   ofstream setC("setC.txt");


   for(int i = 0; i < A; i++){
      email = randomStringWithNum(5) + "." + randomStringWithNum(4) + "@" +
randomStringNoNum(4) + "." + randomDomain();
      if(i == A-1)
         setA << email;
      else
         setA << email << "\n";
   }

   for(int i = 0; i < B; i++){
      email = randomStringWithNum(5) + "." + randomStringWithNum(4) + "@" +
randomStringNoNum(4) + "." + randomDomain();
      if(i == B-1)
         setB << email;
      else
         setB << email << "\n";
   }

   for(int i = 0; i < C; i++){
      email = randomStringWithNum(5) + "." + randomStringWithNum(4) + "@" +
randomStringNoNum(4) + "." + randomDomain();
      if(i == C-1)
         setC << email;
      else
         setC << email << "\n";
   }

   setA.close();
   setB.close();
   setC.close();
}
```

1.3 Explanation of Data Generation

The data generation is fairly simple. The data generated are following a regular expressions given which is [A-Za-z0-9]{5}\.[A-Za-z0-9]{4}@[A-Za-z]{4}\.(com|my|org) . It consists of 3 main algorithms. The first algorithm is to generate a random string with length of 5 with no number inside which is the first part of the regular expression given. The algorithm work in the following sequence:

1. Create a string array with all the letters but no number
2. Create another empty string array which is needed to group the random string
3. Create a integer variable to store the position
4. get a random number from system time and do an integer division of the size of the array consisting of all the letters and store it in the integer variable.
5. Locate the letter of the position of the integer variable inside the string array and add it into the empty array
6. The step 4 and 5 is repeated until the size of the empty array reaches the indicated number.

The second algorithm is to generate a random string with length of 4 with numbers inside. The algorithm work in the following sequence:

1. Create a string array with all the letters including numbers
2. Create another empty string array which is needed to group the random string
3. Create a integer variable to store the position
4. get a random number from system time and do an integer division of the size of the array consisting of all the letters and numbers and store it in the integer variable.
5. Locate the letter of the position of the integer variable inside the string array and add it into the empty array
6. The step 4 and 5 is repeated until the size of the empty array reaches the indicated number.

The third algorithm is needed to complete the email, which is to generate the domain of the email. The algorithm works in the following way:

1. Create a string array with all the possible domain
2. Create another empty string array which is needed to store the random domain
3. get a random number from system time and do an integer division of 3 and store it in the integer variable.
4. Locate the letter of the position of the integer variable inside the array domain and add it into the empty array and return the array

After establishing the algorithms, we can just create an empty string and concatenate the results of the algorithms to create a completely random email and output it into a file.

2.0 Hash Table

2.1 Algorithms Of Hash Table Using Chaining

Algorithm 1 hashFunction

```
1: procedure hashFunction(key)
2:    sum ← 0
3:    for i = 0 until i < keySize do
4:        keyInAscii ← converts key into ASCII integer value
5:        sum += keyInAscii
6:   endFor
7:   return sum % hashTableSize
8: end procedure
```

Algorithm 2 getIndex

```
1: procedure getIndex (key)
2:    index = hashFunction(key)
3:    for i = get hashTable[index] first element  until hashTable[index] last element do
4:        if(i == key)
5:            return index
6:   endFor
7:   return -1
8: end procedure
```

Algorithm 3 getSize

```
1: procedure getSize()
2:   return hashTableSize
3: end procedure
```

Algorithm 4 insertItem

```
1: procedure insertItem(key)
2:     index = hashFunction(key)
3:     hashTable[index].add(key)
4: end procedure
```

Algorithm 5 displayHash()

```
1: procedure displayHash()
2:     for i = 0 until i < hashTableSize do
3:         display i
4:         for x = elements of hashTable until hashTable[i] do
5:             display → and x
6:         endfor
7:     endfor
8: end procedure
```

Algorithm 5 getHashByLine()

```
1: procedure getHashByLine()
4:     for x = elements of hashTable until hashTable[i] do
5:         line = → +  x
6:     endfor
5:     return line
8: end procedure
```

2.2 Program of Hash Table using Chaining

```
#include <fstream>
#include<bits/stdc++.h>
#include <iostream>
#include <string>
#include <chrono>

using namespace std::chrono;
using namespace std;

class Hash
{
    int hashTableSize;
    list<string> *hashTable;

public:
    Hash(int V);
    void insertItem(string key);
    int hashFunction(string key) {
        int sum = 0;
        int index;
        for(int i =0;i<key.length();i++){
            sum += (int)key[i];
        }
        return sum % hashTableSize;
    }
    void displayHash();
    int getIndex(string key);

    int getSize(){
        return (hashTableSize);
    }

    string getHashByLine(int i){
        string line;

        for (auto x : hashTable[i])
            line += " --> " + x  ;

        return line;
    }
};

Hash::Hash(int b)
{
    this->hashTableSize = b;
    hashTable = new list<string>[hashTableSize];
}

void Hash::insertItem(string key)
{
    int index = hashFunction(key);
    hashTable[index].push_back(key);
}


int Hash::getIndex(string key)
{
  // get the hash index of key
  int index = hashFunction(key);

  // find the key in (inex)th list
  list <string> :: iterator i;
  for (i = hashTable[index].begin();
          i != hashTable[index].end(); i++) {
    if (*i == key){
        return index;
    }
  }
  return -1;
}

// function to display hash table
void Hash::displayHash() {
  for (int i = 0; i < hashTableSize; i++) {
    cout << i;
    for (auto x : hashTable[i])
      cout << " --> " << x;
    cout << endl;
  }
}


int main()
{
    ifstream file;
    ofstream result("result.txt");

    int select;
    string key;
    string line;
```

```
int sizeOfHash = 0;


cout << "Please select the file to be read " << endl;
cout << "1. Set A (100 items)" << endl;
cout << "2. Set B (100,000 items)" << endl;
cout << "3. Set C (500,000 items)" << endl;
cout << "Please enter 1,2 or 3 only :";

cin >> select;

if (select == 1){
   file.open("setA.txt");
} else if (select == 2){
   file.open("setB.txt");
}else if (select == 3){
   file.open("setC.txt");
}else {
   cout << "Invalid input !" << endl;
}

while (!file.eof()){
   getline(file,line);
   ++sizeOfHash;
}
file.seekg(0, file.beg);
Hash h(sizeOfHash*0.9);
auto start = high_resolution_clock::now();
while (file >> key){
   h.insertItem(key);
}
auto stop = high_resolution_clock::now();
auto duration = duration_cast<microseconds>(stop - start);


string a;
int sizeH = h.getSize();


h.displayHash();

for(int i =0 ; i < h.getSize() ;i++){
   result <<i <<h.getHashByLine(i) << endl;
}

result << "Time taken to insert value into Hash Table: "
```

```
 << duration.count() << " microseconds" << endl;

 cout << "Time taken to insert value into Hash Table: "
 << duration.count() << " microseconds" << endl;

 file.close();


string fileName;
string fileInput;
int indexOfKey;
while(true){
   cout << "Select the options below " << endl;
   cout << "1. Search data manually " << endl;
   cout << "2. Search data by file input" << endl;
   cout << "Enter 1 or 2 only :";
   cin >> select;

   if(select == 1){
      cout << "Enter the string to find: ";
      cin >> key;
      auto start = high_resolution_clock::now();
      indexOfKey = h.getIndex(key);
      auto stop = high_resolution_clock::now();
      auto duration = duration_cast<microseconds>(stop - start);
      cout << endl;
      if (indexOfKey >= 0){

         cout << "Time taken to find the key in the Hash Table: "
         << duration.count() << " microseconds" << endl;
         cout << "The key is found at index "<< indexOfKey << endl << endl;
      }
      else{
         cout << "The key is not available in the table  "<< endl << endl;
      }
   }
   else if (select == 2){
      cout << "Enter the file name including the file format (eg:found.txt) :";
      cin >> fileName;
      file.open(fileName);
      cout << endl;
      while (file >> fileInput){
         auto start = high_resolution_clock::now();
         indexOfKey = h.getIndex(fileInput);
         auto stop = high_resolution_clock::now();
         auto duration = duration_cast<microseconds>(stop - start);
```

```
            if(indexOfKey >= 0){

                cout << "Time taken to insert value into Hash Table: "
                << duration.count() << " microseconds" << endl;
                cout << "The key is found at key "<< indexOfKey << endl << endl;
            }
            else{
                cout << "The key is not available in the table  "<< endl << endl;
            }
        }
    }
    else{
        cin.clear();
        cin.ignore();
        cout << "invalid input !" <<endl << endl;
        continue;
    }
    file.close();

  }
}
```

2.3 Algorithms Of Hash Table Using Linear Probing

Algorithm 1 hashFunction

```
1: procedure hashFunction(key)
2:     sum ← 0
3:     for i = 0 until i < keySize do
4:         keyInAscii ← converts key into ASCII integer value
5:         sum += keyInAscii
6:     endFor
7:     return sum % hashTableSize
8: end procedure
```

Algorithm 2 getIndex

```
1: procedure getIndex (key)
2:     index = hashFunction(key)
3:     counter ← 0
4:     while hashMap[index] is not invalid do
5:         if(counter++ > hashTableSize) then
6:             break;
7:         if(hashMap[index].email is equal to  key) then
8:             return index;
9:     return -1
```

Algorithm 3 getSize

```
1: procedure getSize()
2:     return hashTableSize
3: end procedure
```

Algorithm 4 insertItem

```
1: procedure insertItem(key)
2:    index = hashFunction(key)
3:    while hashMap[index] is not invalid do
4:        if ++index > hashTableSize then
5:            index = 0
6:    endWhile
7:        hashMap[index] → add in new HashMap with value key
13:  end procedure
```

Algorithm 5 displayHash()

```
1: procedure displayHash()
2:    while true do
3:        if (i > hashTableSize) then
4:            break;
5:        if(hashMap[i] is not invalid) then
6:            display hashMap[i].email
7:   endfor
8: end procedure
```

2.4 Program of Hash Table using Linear Probing

```
#include <fstream>
#include<bits/stdc++.h>
#include <iostream>
#include <string>
#include <typeinfo>
#include <chrono>

using namespace std::chrono;
using namespace std;

class HashMap
{
  public:
    string email;
    HashMap(string email)
    {
      this->email = email;
    }
};

class Hash
{
    int hashTableSize;
    HashMap **hashMap;

public:
    Hash(int V);
    void insertItem(string key);
    int hashFunction(string key) {
        int sum = 0;
        int index;
        for(int i =0;i<key.length();i++){
            sum += (int)key[i];
        }
        return sum % hashTableSize;
    }

    int getIndex(string key);

    int getSize(){
        return (hashTableSize);
    }
    void displayHash();


};

Hash::Hash(int b)
{
    this->hashTableSize = b;
    hashMap = new HashMap*[this->hashTableSize];
}

void Hash::insertItem(string key)
{
    int index = hashFunction(key);

    while(hashMap[index] != NULL)
    {
      if (++index>hashTableSize)
        index = 0;
    }

    hashMap[index] = new HashMap(key);
}


int Hash::getIndex(string key)
{
  int index = hashFunction(key);
  int counter=0;

  while(hashMap[index] != NULL)
  {
      if(counter++ > hashTableSize)
      break;

      if(hashMap[index++]->email == key)
      {
          return index;
      }
  }
  return -1;

}

void Hash::displayHash() {
  int i = 2;
  while(true){
    if (i>hashTableSize){
      break;
    }
    if(hashMap[i] != nullptr){
      cout << (i-1) << "-->" << hashMap[i]->email;
      cout<<endl;
      i++;
    }
    else{
      cout << i << "-->";
      cout<<endl;
      i++;
    }

  }
}



int main()
{
    ifstream file;
    ofstream result("result.txt");

    int select;
    string key;
    string line;
    int sizeOfHash = 0;


    cout << "Please select the file to be read " << endl;
    cout << "1. Set A (100 items)" << endl;
    cout << "2. Set B (100,000 items)" << endl;
    cout << "3. Set C (500,000 items)" << endl;
    cout << "Please enter 1,2 or 3 only :";

    cin >> select;

    if (select == 1){
        file.open("setA.txt");
    } else if (select == 2){
        file.open("setB.txt");
    }else if (select == 3){
        file.open("setC.txt");
```

```
}else {
   cout << "Invalid input !" << endl;
}

while (!file.eof()){
   getline(file,line);
   ++sizeOfHash;
}
file.seekg(0, file.beg);
Hash h(sizeOfHash*1.5);
auto start = high_resolution_clock::now();
while (file >> key){
   h.insertItem(key);
}
auto stop = high_resolution_clock::now();
auto duration = duration_cast<seconds>(stop - start);

if(select == 1 || select == 2){
  h.displayHash();
}




result << "Time taken to insert value into Hash Table: "
 << duration.count() << " seconds" << endl;

 cout << "Time taken to insert value into Hash Table: "
 << duration.count() << " seconds" << endl;

 file.close();


string fileName;
string fileInput;
int indexOfKey;
while(true){
   cout << "Select the options below " << endl;
   cout << "1. Search data manually " << endl;
   cout << "2. Search data by file input" << endl;
   cout << "Enter 1 or 2 only :";
   cin >> select;

   if(select == 1){
      cout << "Enter the string to find: ";
      cin >> key;
```

```
            auto start = high_resolution_clock::now();
            indexOfKey = h.getIndex(key);
            auto stop = high_resolution_clock::now();
            auto duration = duration_cast<seconds>(stop - start);
            cout << endl;
            if (indexOfKey >= 0){

                cout << "Time taken to find the key in the Hash Table: "
                << duration.count() << " seconds" << endl;
                cout << "The key is found at index "<< indexOfKey << endl << endl;
            }
            else{
                cout << "The key is not available in the table  "<< endl << endl;
            }
        }
        else if (select == 2){
            cout << "Enter the file name including the file format (eg:found.txt) :";
            cin >> fileName;
            file.open(fileName);
            cout << endl;
            while (file >> fileInput){
                auto start = high_resolution_clock::now();
                indexOfKey = h.getIndex(fileInput);
                auto stop = high_resolution_clock::now();
                auto duration = duration_cast<microseconds>(stop - start);
                if(indexOfKey >= 0){

                    cout << "Time taken to find the key in the Hash Table: "
                    << duration.count() << " seconds" << endl;
                    cout << "The key is found at index "<< indexOfKey << endl << endl;
                }
                else{
                    cout << "The key " << fileInput << " is not available in the table  "<< endl
<< endl;
                }
            }
        }
        else{
            cin.clear();
            cin.ignore();
            cout << "invalid input !" <<endl << endl;
            continue;
        }
        file.close();
    }
}
```

3.0 AVL Binary Search Tree

3.1 Algorithm of AVL Binary Search Tree

Algorithm 1 rotateRight

```
1: procedure *rotateRight(Node *y)
2:    x ← left child of y
3:    x1 ← right child of y
4:    right child x ← y
5:    left child y ← x1
6:    height of y = height of child +1
7:    height of x = height of child + 1
8:  end procedure
```

Algorithm 2rotateLeft(Node *x)

```
1: procedure *rotateLeft(Node *y)
2:    y ← right child of x
3:    x1 ← left child of y
4:    left child y ← x
5:    right child x ← x1
6:    height of y = height of child +1
7:    height of x = height of child + 1
8:  end procedure
```

Algorithm 3 getBalance(Node *N)

```
1: procedure getBalance(Node *N)
2: if(N does not exist)
3:      Return 0;
4: else
5:     Return (height of left child) - (height of right child)
6:  end procedure
```

Algorithm 3 insert(Node *node, string key)

```
1: procedure insert(Node *node, string key)
2: if(N does not exist)
3:     return new node with key as root
4: if(key < node value)
5:     insert key as left child of node
6: else if(key > node value)
7:     insert key as right child of node
8: else //if key == node value
9:      Return node //no duplicate keys allowed in bst
10: balance ← get balance value of node
11: if(balance < -1 and key > node right child value)
12:      return leftRotate(node);
13: if(balance >1 and key > node left child value)
14:      return rightRotate(node);
15: return node
16: end procedure
```

Algorithm 4 preOrder(Node *root)

```
1: procedure preOrder(Node *root)
2: if(root exist)
3:      cout<< root value
4:      preOrder(root left child)
5:      preorder(root right child)
6: end procedure
```

Algorithm 5 search(Node *root, string key)

```
1: procedure search(Node *root,string key)
2: if(root does not exist)
3:       return 0;
4: else if(root value == key)
5:       return root.height;
6: else if(key <= root value)
7:       return Search(root left child, key)
8: else
9:       return Search(root right child, key)
10: end procedure
```

3.2 Program of AVL Binary Search Tree

```
#include<bits/stdc++.h>
#include<string>
#include<iostream>
#include <fstream>
#include <chrono>
#include <cstdlib>
using namespace std;


class Node
{
    public:
    string key;
    Node *left;
    Node *right;
    int height;
};


int max(int a, int b);

int height(Node *N)
{
    if (N == NULL)
        return 0;
    return N->height;
}

int max(int a, int b)
{
    return (a > b)? a : b;
}


Node* newNode(string key)
{
    Node* node = new Node();
    node->key = key;
    node->left = NULL;
    node->right = NULL;
    node->height = 1; // new node is initially
                      // added at leaf
    return(node);
}


Node *rotateRight(Node *y)
```

```
{
   Node *x = y->left;
   Node *x1 = x->right;

   // Perform rotation
   x->right = y;
   y->left = x1;

   // Update heights
   y->height = max(height(y->left),
                   height(y->right)) + 1;
   x->height = max(height(x->left),
                   height(x->right)) + 1;

   // Return new root
   return x;
}

Node *rotateLeft(Node *x)
{
   Node *y = x->right;
   Node *x1 = y->left;

   // Perform rotation
   y->left = x;
   x->right = x1;

   // Update heights
   x->height = max(height(x->left),
                   height(x->right)) + 1;
   y->height = max(height(y->left),
                   height(y->right)) + 1;

   // Return new root
   return y;
}

// Get Balance factor of node N
int getBalance(Node *N)
{
   if (N == NULL)
      return 0;
   return height(N->left) - height(N->right);
}


Node* insert(Node* node, string key)
{

   if (node == NULL)
```

```
        return(newNode(key));

    if (key < node->key)
        node->left = insert(node->left, key);
    else if (key > node->key)
        node->right = insert(node->right, key);
    else // Equal keys are not allowed in BST
        return node;

    node->height = 1 + max(height(node->left),
                        height(node->right));


    int balance = getBalance(node);


    if (balance > 1 && key < node->left->key)
        return rotateRight(node);


    if (balance < -1 && key > node->right->key)
        return rotateLeft(node);


    if (balance > 1 && key > node->left->key)
    {
        node->left = rotateLeft(node->left);
        return rotateRight(node);
    }


    if (balance < -1 && key < node->right->key)
    {
        node->right = rotateRight(node->right);
        return rotateLeft(node);
    }


    return node;
}

void preOrder(Node *root)
{
    if(root != NULL)
    {
        //cout << "\n";
        //cout << height(root) << "\n";
        cout << root->key << "  -->" ;
        preOrder(root->left);
        preOrder(root->right);
    }
    //cout << root->key << "End of tree" ;
}

//search in Preorder (Root, Left, Right) : 1 2 4 5 3
int Search(Node* root,string key) {
        if(root == NULL) {
                return 0;
        }
        else if(root->key == key) {
                return root->height;
        }
        else if(key <= root->key) {
                return Search(root->left,key);
        }
        else {
                return Search(root->right,key);
        }
}

int main()
{
    Node *root = NULL;
    string myText;
    string file;
    string choice;
    int index;

    cout<<"Choose a dataset A, B, C\n";
        cin>>choice;

    if (choice.compare("A") == 0){
        file = "setA.txt";
    }
    else if(choice.compare("B") == 0){
        file = "setB.txt";
    }
    else{
        file = "setC.txt";
    }

    ifstream MyReadFile(file);
    auto start = chrono::system_clock::now();
    while (getline (MyReadFile, myText)) {
        root = insert(root,myText);
    }

    auto end = chrono::system_clock::now();
    chrono::duration<double> duration = end - start;
    cout << "Insert Duration: " << duration.count() << "s\n";
    MyReadFile.close();

    string text;
    cout<<"Enter an email be searched\n";
            cin>>text;
    auto start1 = chrono::system_clock::now();
    index = Search(root,text);
    if(index != 0)
    {
        cout<<"Found at height: ";
        cout<< index;
        cout<< '\n';
    }
    else cout<<"Not Found\n";

    auto end1 = chrono::system_clock::now();
    chrono::duration<double> duration1 = end1 - start1;
    cout << "Search Duration: " << duration1.count() << "s\n";
    cout << "Show binary tree sequence? (y/n): " << "\n";
    string show;
    cin >> show;
    if(show.compare("y") == 0)
    {
        preOrder(root);
        cout << "end of tree";
    }
    return 0;
}
```

4.0 Priority queue using Heap

4.1 Algorithm of Priority queue using Heap

Algorithm 1 Enqueue

```
1:  procedure Enqueue(element)
2:      Insert element to the end of queue
3:      HeapifyEnqueue(last location of queue)
4:  end procedure
```

Algorithm 2 HeapifyEnqueue

```
1:  procedure HeapifyEnqueue(index)
2:      if index == 0 then
3:          return
4:      end if
5:      parent ← (index -1)/2
6:      if Array[index] > Array[parent] then
7:          swap(Array[index] , Array[parent])
8:          HeapifyEnqueue(parent)
9:      end if
10: end procedure
```

Algorithm 3 Dequeue

```
1:  procedure Dequeue
2:      removedElement ← Array[0]
3:      Array[0] last element of Array
4:      Remove last element of Array
5:      HeapifyDequeue(0)
6:      return removedElement
7:  end procedure
```

Algorithm 4 HeapifyDequeue

```
1: procedure HeapifyDequeue(index)
2:     leftChild ← (2 * index ) + 1
3:     rightChild ← (2 * index ) + 2
4:     if rightChild < Arraysize then
5:         if Array[leftChild] > Array[rightChild] then
6:             max ←leftChild
7:         else
8:             max ←rightChild
9:         endif
10:        if Array[index] > Array[max] then
11:            max ←index
12:        endif
13:    else if leftChild < Arraysize then
14:        if Array[leftChild] > Array[index] then
15:            max ←leftChild
16:        else
17:            max ←index
18:        end if
19:    else
20:        max ←index
21:    endif
22:    if max != index then
23:        swap(Array[max], Array[index])
24:        HeapifyDequeue(max)
25:    endif
26: end procedure
```

Algorithm 5 size

```
1: procedure size( )
2:     return Arraysize
3: end procedure
```

Algorithm 6 print

```
1: procedure print( )
2:     for i = 0 until i < Arraysize do
3:         display i
4:     endfor
5: end procedure
```

4.2 Program of PriorityQueue using heap

```
#include <fstream>
#include<bits/stdc++.h>
#include <iostream>
#include <string>
#include <chrono>
#include <stdlib.h>

using namespace std::chrono;
using namespace std;


template <typename T>
class PriorityQueue{
    vector<T> A;

    void heapify_enqueue(int index){
        if(index == 0){
            return ;
        }

        //parent index
        int parent = (index-1)/2;

        //swap if parent is smaller;
        if(A[index].compare(A[parent]) > 0){
            swap(A[index], A[parent]);
            //recursion of the function
            heapify_enqueue(parent);
        }
    }

    void heapify_dequeue(int index){
        int max;

        //left child index
        int leftChild = ((2 * index) + 1);

        //right child index
        int rightChild = ((2 * index) + 2);

        // compare and find the greatest child
        if(rightChild < A.size()){
            if(A[leftChild].compare(A[rightChild]) > 0){
                max = leftChild;
            }else{
                max = rightChild;
            }
            if(A[index].compare(A[max]) > 0){
                max = index;
            }

        }else if(leftChild < A.size()){
            if(A[leftChild].compare(A[index]) > 0){
                if(A[leftChild].compare(A[index]) > 0){
                    max = leftChild;
                }else{
                    max = index;
                }
            }

        }else{
            max = index;
        }

        if(max != index){
            swap(A[index],A[max]);
            heapify_dequeue(max);
        }
    }

    public:
        void enqueue(T element){
            A.push_back(element);
            heapify_enqueue(A.size()-1);
        }

        T dequeue(){
            T removed_element = A[0];
            A[0] = A[A.size()-1];
            A.pop_back();
            heapify_dequeue(0);
            return removed_element;
        }

        int size(){
            return A.size();
        }

        void print(){
            for(int i=0; i<A.size(); i++){
                cout << A[i] << " | ";
            }
            cout << endl;
        }
};
```

```
int main(){
    ifstream file;
    ofstream result("result.txt");

    int select;
    string key;
    string line;
    bool status = true;
    int sizeOfPQ = 0;

    while(true){
        cout << "+++++++++++++++++++++++++++++++++++++" <<endl;
        cout << "+" << " Please select the file to be read " <<"+"<< endl;
        cout << "+++++++++++++++++++++++++++++++++++++" <<endl;
        cout << "+" << " 1. Set A (100 items)" << "             +" << endl;
        cout << "+" << " 2. Set B (100,000 items)" << "         + " << endl;
        cout << "+" << " 3. Set C (500,000 items)" << "         + " << endl;
        cout << "+++++++++++++++++++++++++++++++++++++" <<endl;
        cout << endl;
        cout << "Please enter 1,2 or 3 only : " ;


        cin >> select;

        if (select == 1){
            file.open("setA.txt");
        } else if (select == 2){
            file.open("setB.txt");
        }else if (select == 3){
            file.open("setC.txt");
        }else {
            cout << "Invalid input !" << endl;
            exit(3);
            return 3;
        }

        while (!file.eof()){
            getline(file,line);
        }
        file.seekg(0, file.beg);

        PriorityQueue<string> pq;

        auto start_enq = high_resolution_clock::now();
        while (file >> key){
            pq.enqueue (key);
        }
        auto stop_enq = high_resolution_clock::now();
        auto duration_enq = duration_cast<microseconds>(stop_enq - start_enq);
```

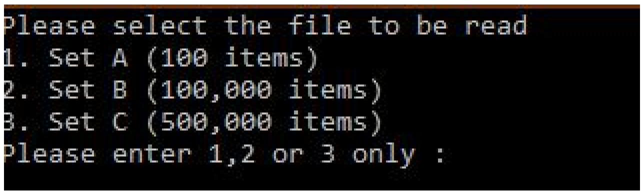
```
	file.close();

	int after_enq_size = pq.size();
	int ten_percent = after_enq_size*0.1;

	//print enqueue
	cout << endl;
	cout << "PriorityQueue after enqueue: " << endl;
	cout << endl;
	pq.print();
	cout << endl;

	//Dequeue 10 % of data
	cout << "Dequeue the first element of the priority queue line by line:" << endl;
	cout << endl;
	auto start_deq = high_resolution_clock::now();
	for(int i = 0; i < ten_percent; i++){
	   cout << pq.dequeue() << endl;
	}
	auto stop_deq = high_resolution_clock::now();
	auto duration_deq = duration_cast<microseconds>(stop_deq - start_deq);

	int after_deq_size = pq.size();

	//print dequeue
	cout << endl;
	cout << "PriorityQueue after dequeue: " << endl;
	cout << endl;
	pq.print();
	cout << endl;


	cout << "\nEnqueue\t: PriorityQueue\n";
	result << "Time taken to enqueue into Priority Priority Queue: "
	<< duration_enq.count() << " microseconds" << endl;

	cout << endl;
	cout << "There are " << after_enq_size << " elements in the priority queue after
enqueue." << endl;
	cout << endl;
	cout << "Time taken to enqueue into Priority Priority Queue: "
	<< duration_enq.count() << " microseconds" << endl;

	cout << "\nDequeue\t: PriorityQueue\n";
	result << "Time taken to dequeue value into Priority Priority Queue: "
	<< duration_deq.count() << " microseconds" << endl;

	cout << endl;
	cout << "There are " << after_deq_size << " elements in the priority queue after
```

```
dequeue." << endl;
        cout << endl;
        cout << "Time taken to dequeue value into Priority Queue: "
        << duration_deq.count() << " microseconds" << endl;

        cout << " " <<endl;

    }

}
```

5.0 Experimental Results

5.1 Hash Table

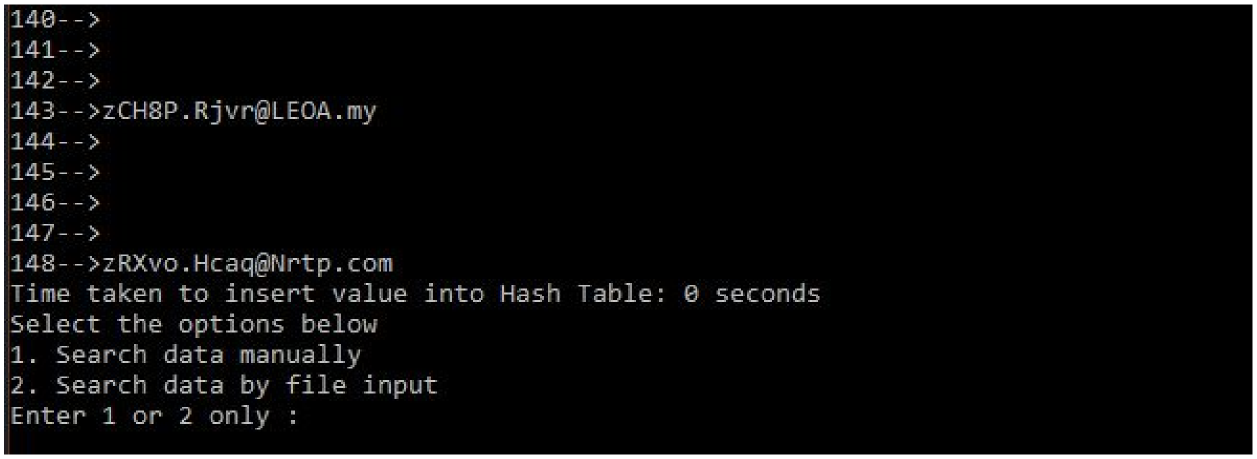
Menu for both Hash Table



```
Please select the file to be read
1. Set A (100 items)
2. Set B (100,000 items)
3. Set C (500,000 items)
Please enter 1,2 or 3 only :
```

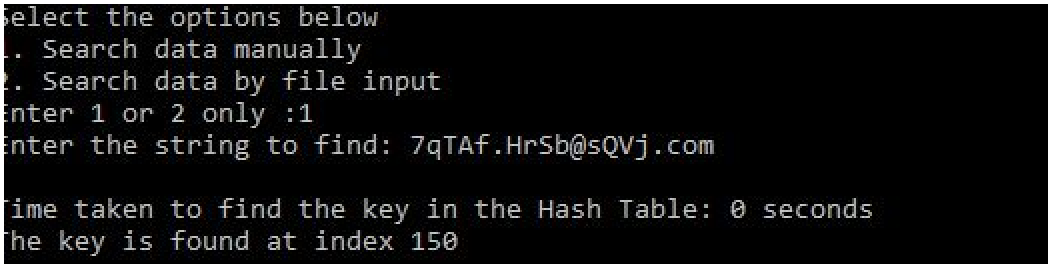
5.1.1 Hash Table with Chaining

5.1.1.1 Set A Results:



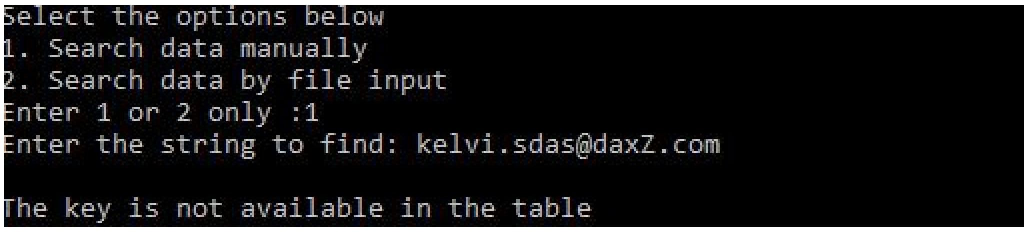
```
140-->
141-->
142-->
143-->zCH8P.Rjvr@LEOA.my
144-->
145-->
146-->
147-->
148-->zRXvo.Hcaq@Nrtp.com
Time taken to insert value into Hash Table: 0 seconds
Select the options below
1. Search data manually
2. Search data by file input
Enter 1 or 2 only :
```

Searching email "7qTAf.HrSb@sQVj.com" which is found in set A



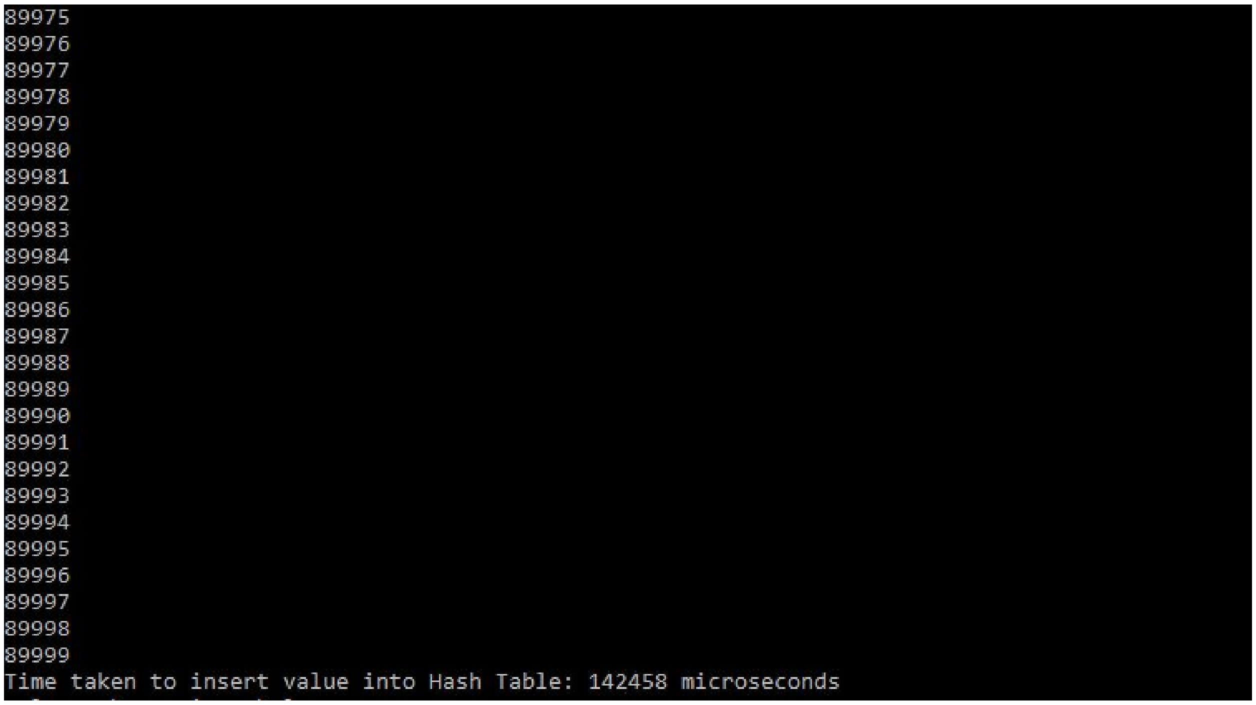
```
elect the options below
. Search data manually
. Search data by file input
nter 1 or 2 only :1
nter the string to find: 7qTAf.HrSb@sQVj.com

ime taken to find the key in the Hash Table: 0 seconds
he key is found at index 150
```

Searching email “kelvi.sdas@daxZ.com “ which is not found in set A:



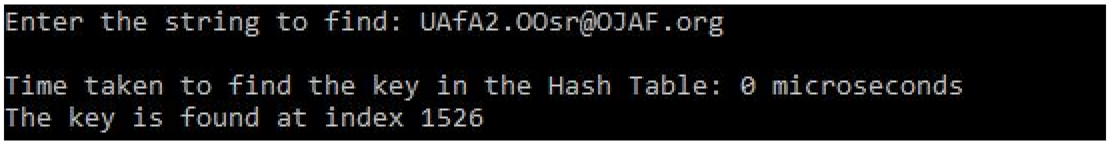
```
Select the options below
1. Search data manually
2. Search data by file input
Enter 1 or 2 only :1
Enter the string to find: kelvi.sdas@daxZ.com

The key is not available in the table
```

5.1.1.2 Set B Results :



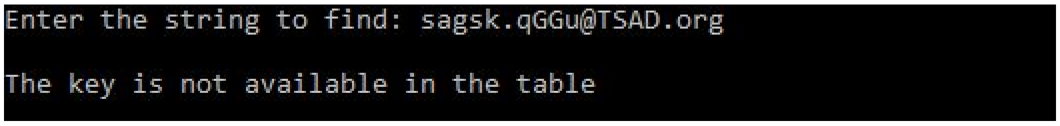
```
89975
89976
89977
89978
89979
89980
89981
89982
89983
89984
89985
89986
89987
89988
89989
89990
89991
89992
89993
89994
89995
89996
89997
89998
89999
Time taken to insert value into Hash Table: 142458 microseconds
```

Searching email “UAfA2.OOsr@OJAF.org” which is found in set B:



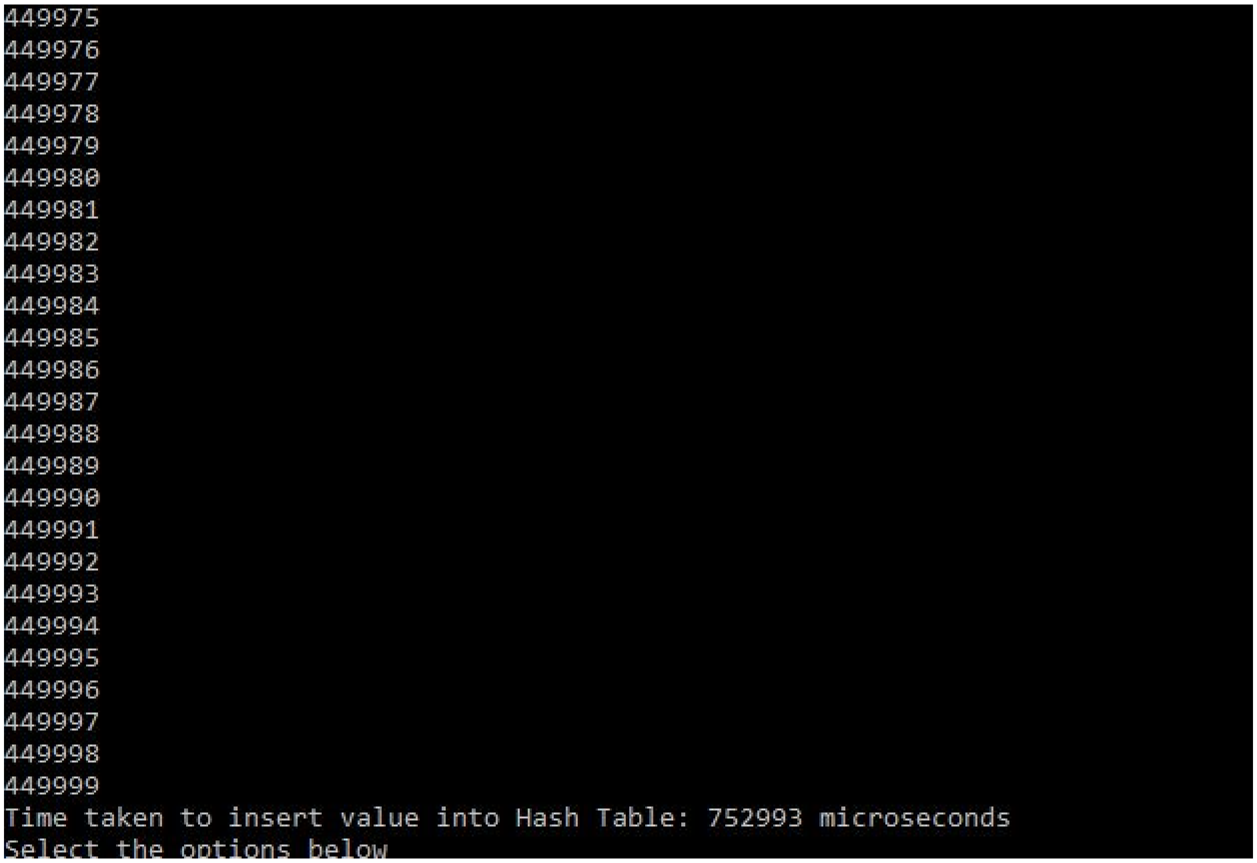
```
Enter the string to find: UAfA2.OOsr@OJAF.org

Time taken to find the key in the Hash Table: 0 microseconds
The key is found at index 1526
```

Searching email “<u>sagsk.qGGu@TSAD.org</u>” which is not found in set B:



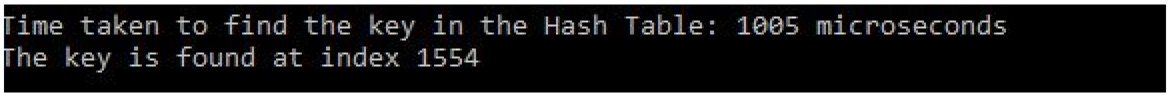
```
Enter the string to find: sagsk.qGGu@TSAD.org

The key is not available in the table
```

5.1.1.3 Set C Results :



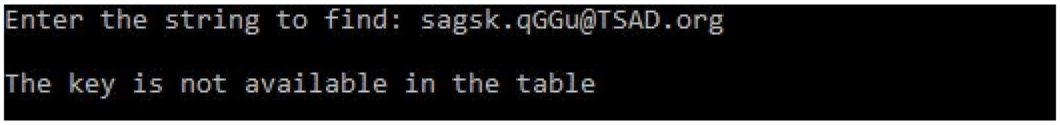
```
449975
449976
449977
449978
449979
449980
449981
449982
449983
449984
449985
449986
449987
449988
449989
449990
449991
449992
449993
449994
449995
449996
449997
449998
449999
Time taken to insert value into Hash Table: 752993 microseconds
Select the options below
```

Searching email “LkKXd.gF3k@DMSA.org” which is found in set C:



```
Time taken to find the key in the Hash Table: 1005 microseconds
The key is found at index 1554
```

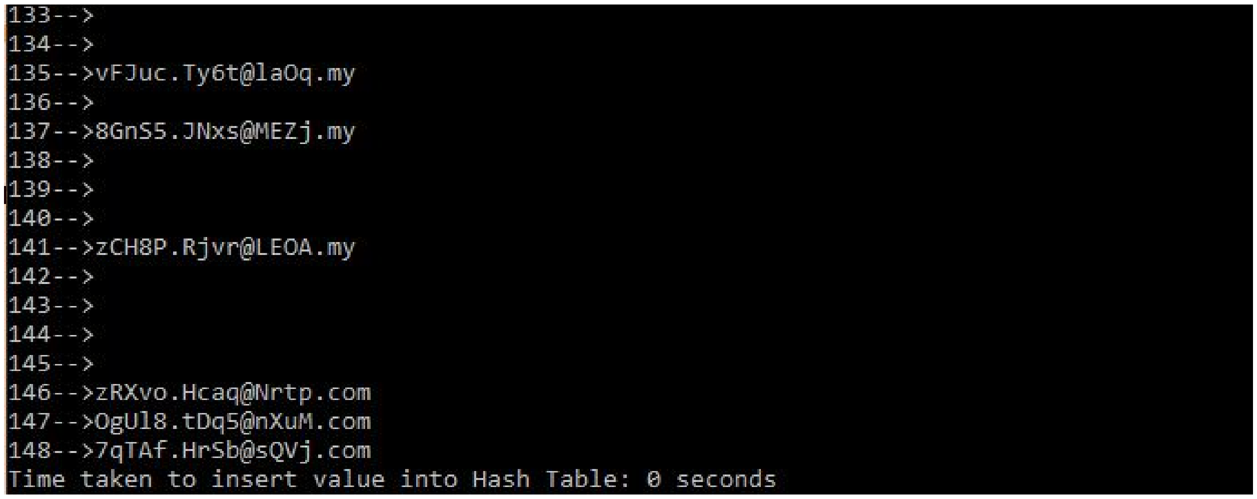
Searching email “sagsk.qGGu@TSAD.org” which is not found in set C:



```
Enter the string to find: sagsk.qGGu@TSAD.org

The key is not available in the table
```

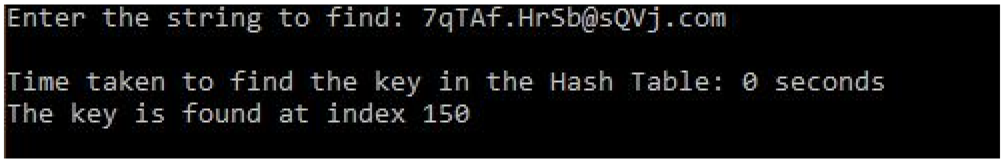
5.1.2 Hash Table with Linear Probing

5.1.2.1 Set A Results :



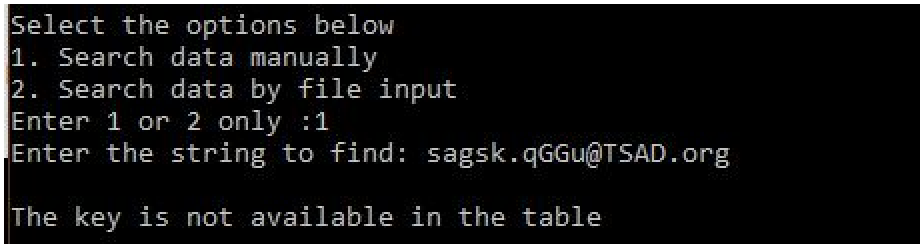
```
133-->
134-->
135-->vFJuc.Ty6t@laOq.my
136-->
137-->8GnS5.JNxs@MEZj.my
138-->
139-->
140-->
141-->zCH8P.Rjvr@LEOA.my
142-->
143-->
144-->
145-->
146-->zRXvo.Hcaq@Nrtp.com
147-->OgUl8.tDq5@nXuM.com
148-->7qTAf.HrSb@sQVj.com
Time taken to insert value into Hash Table: 0 seconds
```

Searching email “7qTAf.HrSb@sQVj.com” which is found in set A:



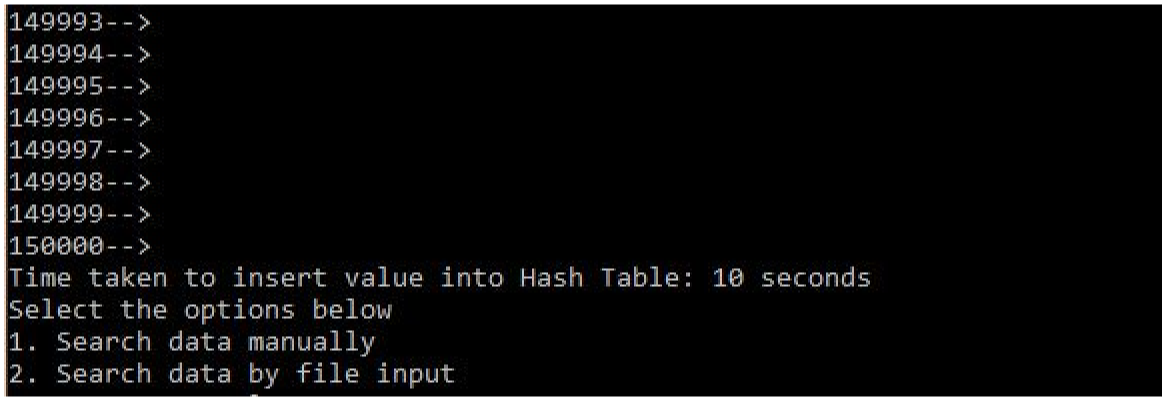
```
Enter the string to find: 7qTAf.HrSb@sQVj.com

Time taken to find the key in the Hash Table: 0 seconds
The key is found at index 150
```

Searching email “sagsk.qGGu@TSAD.org“ which is not found in set A:



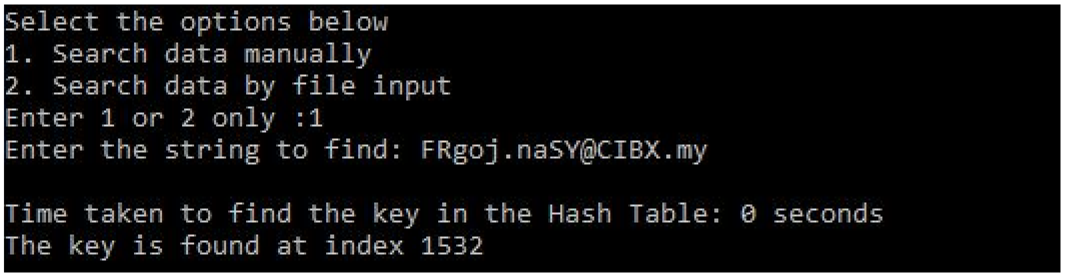
```
Select the options below
1. Search data manually
2. Search data by file input
Enter 1 or 2 only :1
Enter the string to find: sagsk.qGGu@TSAD.org

The key is not available in the table
```

5.1.2.2 Set B Results :



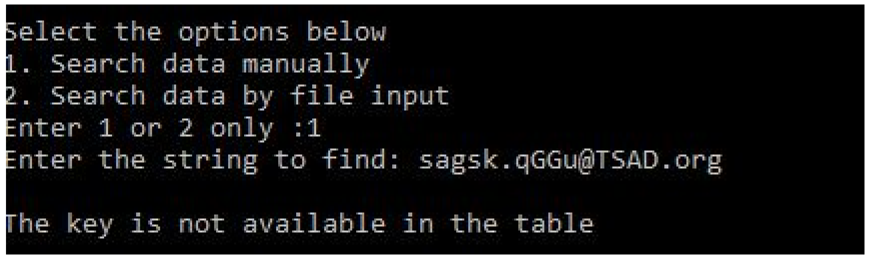
```
149993-->
149994-->
149995-->
149996-->
149997-->
149998-->
149999-->
150000-->
Time taken to insert value into Hash Table: 10 seconds
Select the options below
1. Search data manually
2. Search data by file input
```

Searching email “FRgoj.naSY@CIBX.my” which is found in set B:



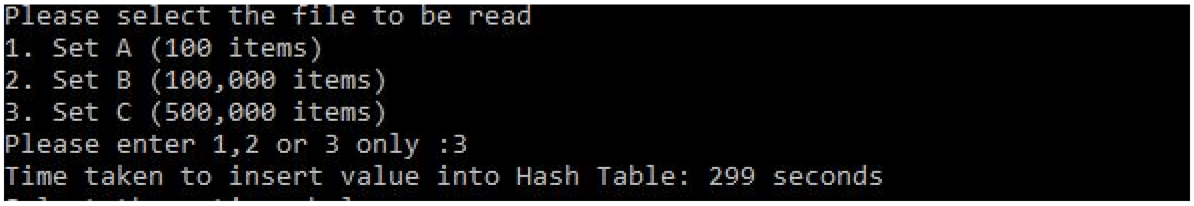
```
Select the options below
1. Search data manually
2. Search data by file input
Enter 1 or 2 only :1
Enter the string to find: FRgoj.naSY@CIBX.my

Time taken to find the key in the Hash Table: 0 seconds
The key is found at index 1532
```

Searching email “sagsk.qGGu@TSAD.org“ which is not found in set B:



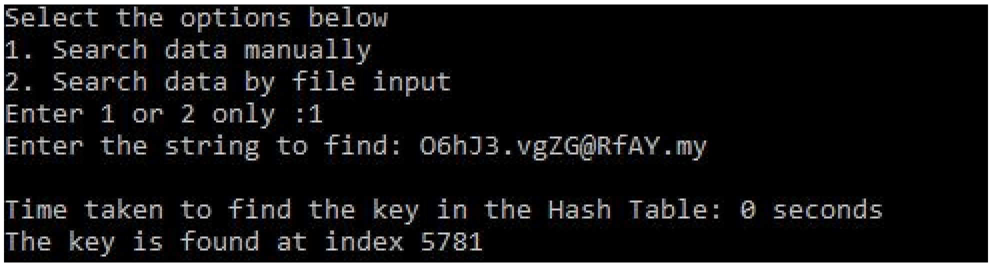
```
Select the options below
1. Search data manually
2. Search data by file input
Enter 1 or 2 only :1
Enter the string to find: sagsk.qGGu@TSAD.org

The key is not available in the table
```

5.1.2.3 Set C Results :



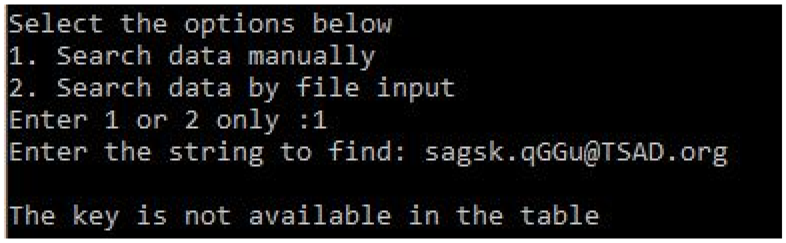
```
Please select the file to be read
1. Set A (100 items)
2. Set B (100,000 items)
3. Set C (500,000 items)
Please enter 1,2 or 3 only :3
Time taken to insert value into Hash Table: 299 seconds
```

Searching email “O6hJ3.vgZG@RfAY.my” which is found in set C:



```
Select the options below
1. Search data manually
2. Search data by file input
Enter 1 or 2 only :1
Enter the string to find: O6hJ3.vgZG@RfAY.my

Time taken to find the key in the Hash Table: 0 seconds
The key is found at index 5781
```

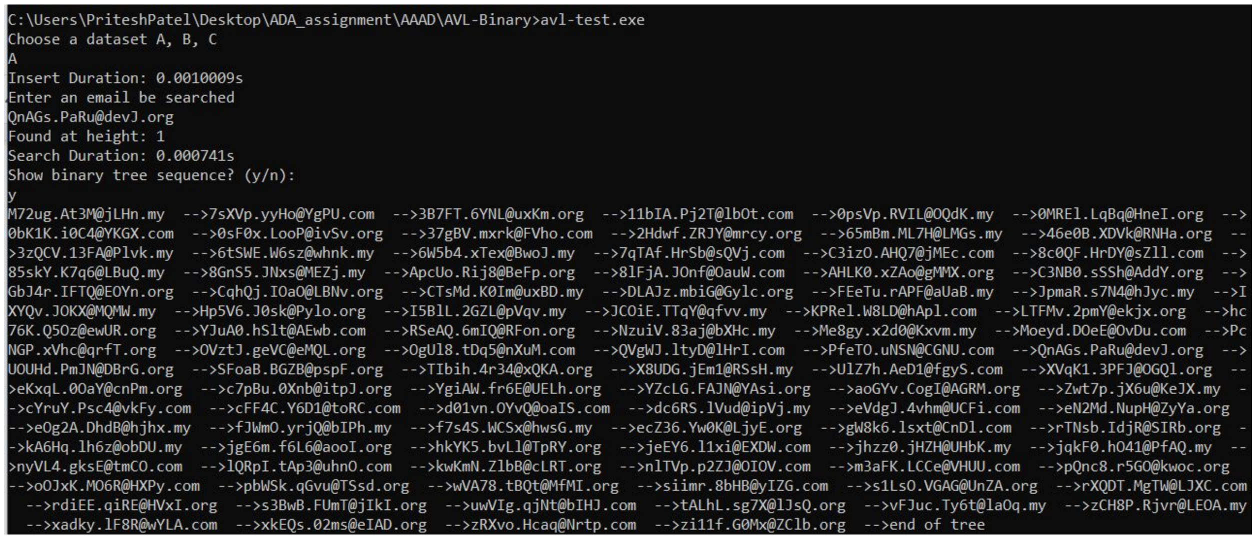
Searching email “sagsk.qGGu@TSAD.org“ which is not found in set C:



```
Select the options below
1. Search data manually
2. Search data by file input
Enter 1 or 2 only :1
Enter the string to find: sagsk.qGGu@TSAD.org

The key is not available in the table
```

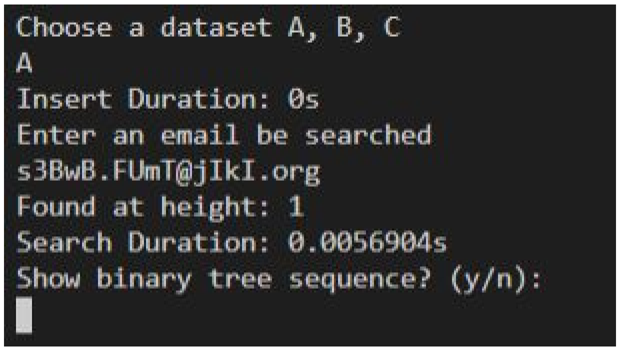
5.2 AVL Binary Search Tree

5.2.1 Set A



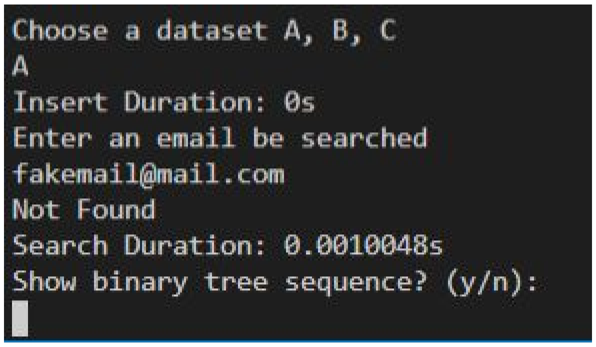
```
C:\Users\PriteshPatel\Desktop\ADA_assignment\AAAD\AVL-Binary>avl-test.exe
Choose a dataset A, B, C
A
Insert Duration: 0.0010009s
Enter an email be searched
QnAGs.PaRu@devJ.org
Found at height: 1
Search Duration: 0.000741s
Show binary tree sequence? (y/n):
y
M72ug.At3M@jLHn.my  -->7sXVp.yyHo@YgPU.com  -->3B7FT.6YNL@uxKm.org  -->11bIA.Pj2T@lbOt.com  -->0psVp.RVIL@OQdK.my  -->0MREl.LqBq@HneI.org  -->
0bK1K.i0C4@YKGX.com  -->0sF0x.LooP@ivSv.org  -->37gBV.mxrk@FVho.com  -->2Hdwf.ZRJY@mrcy.org  -->65mBm.ML7H@LMGs.my  -->46e0B.XDVk@RNHa.org  --
>3zQCV.13FA@Plvk.my  -->6tSWE.W6sz@whnk.my  -->6W5b4.xTex@BwoJ.my  -->7qTAf.HrSb@sQVj.com  -->C3izO.AHQ7@jMEc.com  -->8c0QF.HrDY@sZll.com  -->
85skY.K7q6@LBuQ.my  -->8GnS5.JNxs@MEZj.my  -->ApcUo.Rij8@BeFp.org  -->8lFjA.JOnf@OauW.com  -->AHLK0.xZAo@gMMX.org  -->C3NB0.sSSh@AddY.org  -->
GbJ4r.IFTQ@EOYn.org  -->CqhQj.IOaO@LBNv.org  -->CTsMd.K0Im@uxBD.my  -->DLAJz.mbiG@Gylc.org  -->FEeTu.rAPF@aUaB.my  -->JpmaR.s7N4@hJyc.my  -->I
XYQv.JOKX@MQMW.my  -->Hp5V6.J0sk@Pylo.org  -->I5BlL.2GZL@pVqv.my  -->JCOiE.TTqY@qfvv.my  -->KPRel.W8LD@hApl.com  -->LTFMv.2pmY@ekjx.org  -->hc
76K.Q5Oz@ewUR.org  -->YJuA0.hSlt@AEwb.com  -->RSeAQ.6mIQ@RFon.org  -->NzuiV.83aj@bXHc.my  -->Me8gy.x2d0@Kxvm.my  -->Moeyd.DOeE@OvDu.com  -->Pc
NGP.xVhc@qrfT.org  -->OVztJ.geVC@eMQL.org  -->OgUl8.tDq5@nXuM.com  -->QVgWJ.ltyD@lHrI.com  -->PfeTO.uNSN@CGNU.com  -->QnAGs.PaRu@devJ.org  -->
UOUHd.PmJN@DBrG.org  -->SFoaB.BGZB@pspF.org  -->TIbih.4r34@xQKA.org  -->X8UDG.jEm1@RSsH.my  -->UlZ7h.AeD1@fgyS.com  -->XVqK1.3PFJ@OGQl.org  --
>eKxqL.0OaY@cnPm.org  -->c7pBu.0Xnb@itpJ.org  -->YgiAW.fr6E@UELh.org  -->YZcLG.FAJN@YAsi.org  -->aoGYv.CogI@AGRM.org  -->Zwt7p.jX6u@KeJX.my  -
->cYruY.Psc4@vkFy.com  -->cFF4C.Y6D1@toRC.com  -->d01vn.OYvQ@oaIS.com  -->dc6RS.lVud@ipVj.my  -->eVdgJ.4vhm@UCFi.com  -->eN2Md.NupH@ZyYa.org
-->eOg2A.DhdB@hjhx.my  -->fJWmO.yrjQ@bIPh.my  -->f7s4S.WCSx@hwsG.my  -->ecZ36.Yw0K@LjyE.org  -->gW8k6.lsxt@CnDl.com  -->rTNsb.IdjR@SIRb.org  -
->kA6Hq.lh6z@obDU.my  -->jgE6m.f6L6@aooI.org  -->hkYK5.bvLl@TpRY.org  -->jeEY6.l1xi@EXDW.com  -->jhzz0.jHZH@UHbK.my  -->jqkF0.hO41@PfAQ.my  --
>nyVL4.gksE@tmCO.com  -->lQRpI.tAp3@uhnO.com  -->kwKmN.ZlbB@cLRT.org  -->nlTVp.p2ZJ@OIOV.com  -->m3aFK.LCCe@VHUU.com  -->pQnc8.r5GO@kwoc.org
-->oOJxK.MO6R@HXPy.com  -->pbWSk.qGvu@TSsd.org  -->wVA78.tBQt@MfMI.org  -->siimr.8bHB@yIZG.com  -->s1LsO.VGAG@UnZA.org  -->rXQDT.MgTW@LJXC.com
  -->rdiEE.qiRE@HVxI.org  -->s3BwB.FUmT@jIkI.org  -->uwVIg.qjNt@bIHJ.com  -->tALhL.sg7X@lJsQ.org  -->vFJuc.Ty6t@laOq.my  -->zCH8P.Rjvr@LEOA.my
  -->xadky.lF8R@wYLA.com  -->xkEQs.02ms@eIAD.org  -->zRXvo.Hcaq@Nrtp.com  -->zi11f.G0Mx@ZClb.org  -->end of tree
```

Searching email “s3BwB.FUmT@jIkI.org” which is found in set A:



```
Choose a dataset A, B, C
A
Insert Duration: 0s
Enter an email be searched
s3BwB.FUmT@jIkI.org
Found at height: 1
Search Duration: 0.0056904s
Show binary tree sequence? (y/n):
```

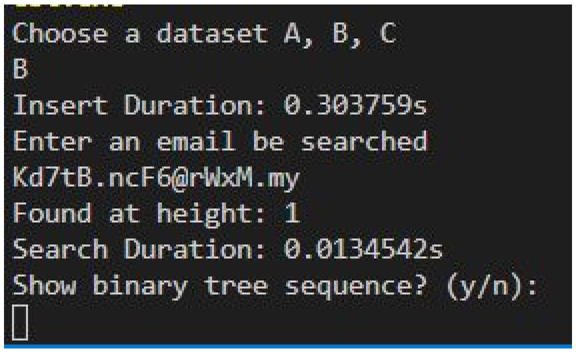
Searching email “fakemail@mail.com” which is not found in set A:



```
Choose a dataset A, B, C
A
Insert Duration: 0s
Enter an email be searched
fakemail@mail.com
Not Found
Search Duration: 0.0010048s
Show binary tree sequence? (y/n):
```

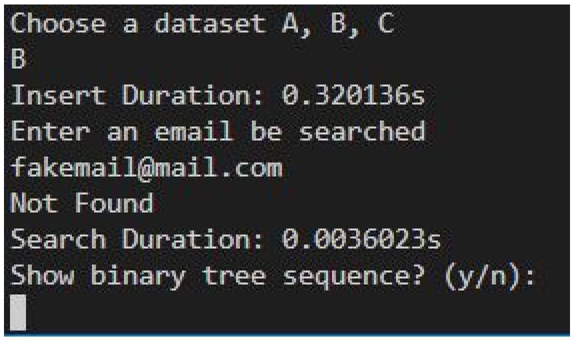
5.2.2 Set B

Searching email “Kd7tB.ncF6@rWxM.my” which is found in set B:



```
Choose a dataset A, B, C
B
Insert Duration: 0.303759s
Enter an email be searched
Kd7tB.ncF6@rWxM.my
Found at height: 1
Search Duration: 0.0134542s
Show binary tree sequence? (y/n):
```

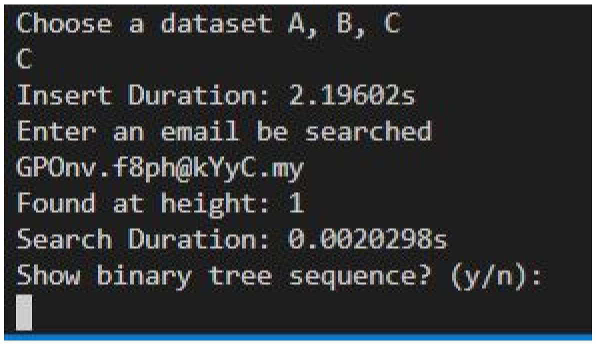
Searching email “fakemail@mail.com” which is not found in set B:



```
Choose a dataset A, B, C
B
Insert Duration: 0.320136s
Enter an email be searched
fakemail@mail.com
Not Found
Search Duration: 0.0036023s
Show binary tree sequence? (y/n):
```

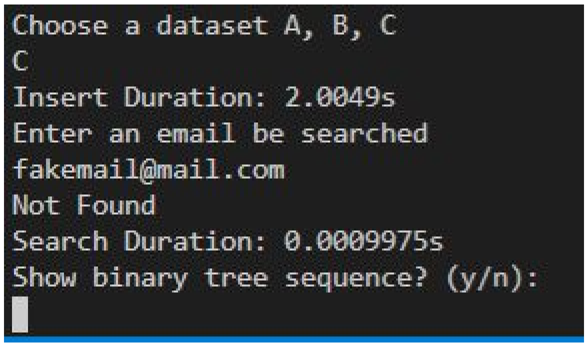
5.2.3 Set C

Searching email "GPOnv.f8ph@kYyC.my" which is found in set C:



```
Choose a dataset A, B, C
C
Insert Duration: 2.19602s
Enter an email be searched
GPOnv.f8ph@kYyC.my
Found at height: 1
Search Duration: 0.0020298s
Show binary tree sequence? (y/n):
```

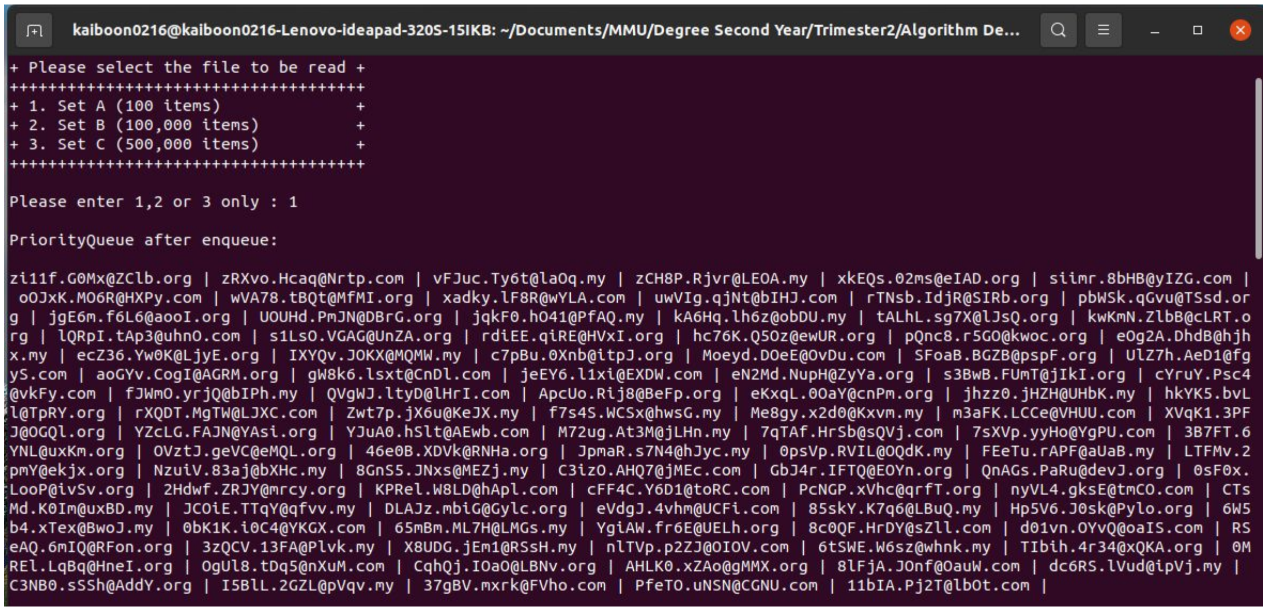
Searching email "fakemail@mail.com" which is not found in set C:



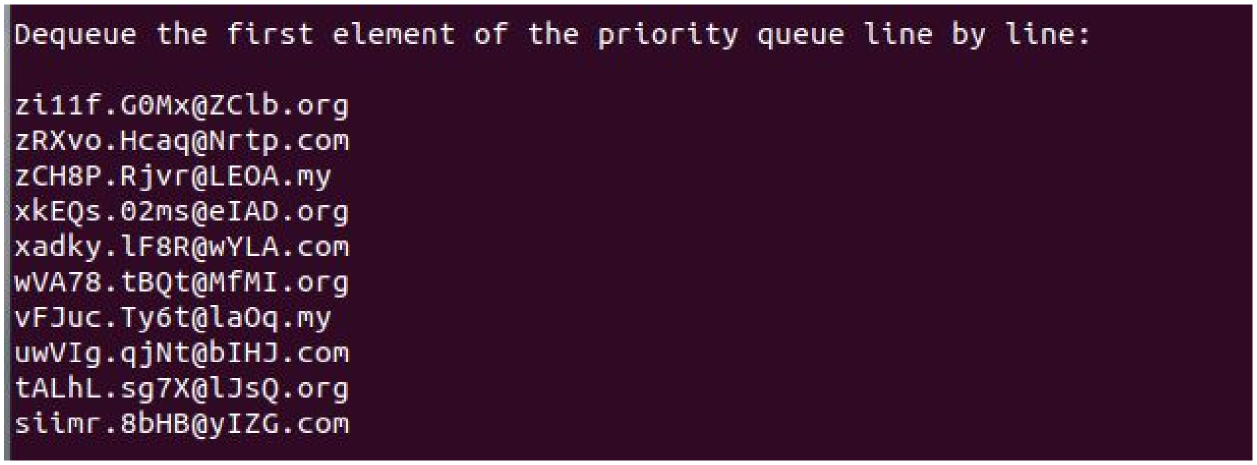
```
Choose a dataset A, B, C
C
Insert Duration: 2.0049s
Enter an email be searched
fakemail@mail.com
Not Found
Search Duration: 0.0009975s
Show binary tree sequence? (y/n):
```

5.3 Priority Queue using Heap

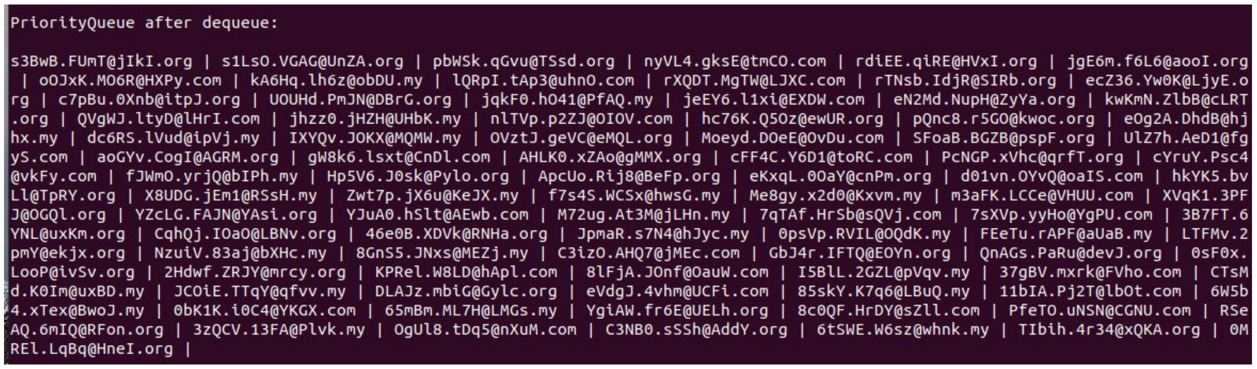
5.3.1 Set A



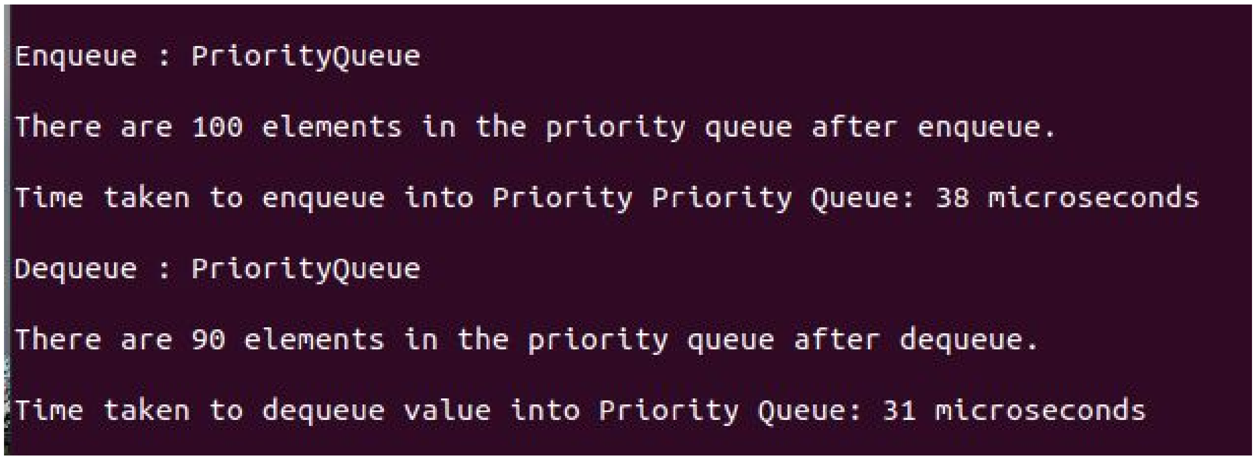
```
kaiboon0216@kaiboon0216-Lenovo-ideapad-320S-15IKB: ~/Documents/MMU/Degree Second Year/Trimester2/Algorithm De...
+ Please select the file to be read +
++++++++++++++++++++++++++++++++++++++
+ 1. Set A (100 items)               +
+ 2. Set B (100,000 items)           +
+ 3. Set C (500,000 items)           +
++++++++++++++++++++++++++++++++++++++

Please enter 1,2 or 3 only : 1

PriorityQueue after enqueue:

zi11f.G0Mx@ZClb.org | zRXvo.Hcaq@Nrtp.com | vFJuc.Ty6t@laOq.my | zCH8P.Rjvr@LEOA.my | xkEQs.02ms@eIAD.org | siimr.8bHB@yIZG.com |
 oOJxK.MO6R@HXPy.com | wVA78.tBQt@MfMI.org | xadky.lF8R@wYLA.com | uwVIg.qjNt@bIHJ.com | rTNsb.IdjR@SIRb.org | pbWSk.qGvu@TSsd.or
g | jgE6m.f6L6@aooI.org | UOUHd.PmJN@DBrG.org | jqkF0.hO41@PfAQ.my | kA6Hq.lh6z@obDU.my | tALhL.sg7X@lJsQ.org | kwKmN.ZlbB@cLRT.o
rg | lQRpI.tAp3@uhnO.com | s1LsO.VGAG@UnZA.org | rdiEE.qiRE@HVxI.org | hc76K.Q5Oz@ewUR.org | pQnc8.r5GO@kwoc.org | eOg2A.DhdB@hjh
x.my | ecZ36.Yw0K@LjyE.org | IXYQv.JOKX@MQMW.my | c7pBu.0Xnb@itpJ.org | Moeyd.DOeE@OvDu.com | SFoaB.BGZB@pspF.org | UlZ7h.AeD1@fg
yS.com | aoGYv.CogI@AGRM.org | gW8k6.lsxt@CnDl.com | jeEY6.l1xi@EXDW.com | eN2Md.NupH@ZyYa.org | s3BwB.FUmT@jIkI.org | cYruY.Psc4
@vkFy.com | fJWmO.yrjQ@bIPh.my | QVgWJ.ltyD@lHrI.com | ApcUo.Rij8@BeFp.org | eKxqL.0OaY@cnPm.org | jhzz0.jHZH@UHbK.my | hkYK5.bvL
l@TpRY.org | rXQDT.MgTW@LJXC.com | Zwt7p.jX6u@KeJX.my | f7s4S.WCSx@hwsG.my | Me8gy.x2d0@Kxvm.my | m3aFK.LCCe@VHUU.com | XVqK1.3PF
J@OGQl.org | YZcLG.FAJN@YAsi.org | YJuA0.hSlt@AEwb.com | M72ug.At3M@jLHn.my | 7qTAf.HrSb@sQVj.com | 7sXVp.yyHo@YgPU.com | 3B7FT.6
YNL@uxKm.org | OVztJ.geVC@eMQL.org | 46e0B.XDVk@RNHa.org | JpmaR.s7N4@hJyc.my | 0psVp.RVIL@OQdK.my | FEeTu.rAPF@aUaB.my | LTFMv.2
pmY@ekjx.org | NzuiV.83aj@bXHc.my | 8GnS5.JNxs@MEZj.my | C3izO.AHQ7@jMEc.com | GbJ4r.IFTQ@EOYn.org | QnAGs.PaRu@devJ.org | 0sF0x.
LooP@ivSv.org | 2Hdwf.ZRJY@mrcy.org | KPRel.W8LD@hApl.com | cFF4C.Y6D1@toRC.com | PcNGP.xVhc@qrfT.org | nyVL4.gksE@tmCO.com | CTs
Md.K0Im@uxBD.my | JCOiE.TTqY@qfvv.my | DLAJz.mbiG@Gylc.org | eVdgJ.4vhm@UCFi.com | 85skY.K7q6@LBuQ.my | Hp5V6.J0sk@Pylo.org | 6W5
b4.xTex@BwoJ.my | 0bK1K.i0C4@YKGX.com | 65mBm.ML7H@LMGs.my | YgiAW.fr6E@UELh.org | 8c0QF.HrDY@sZll.com | d01vn.OYvQ@oaIS.com | RS
eAQ.6mIQ@RFon.org | 3zQCV.13FA@Plvk.my | X8UDG.jEm1@RSsH.my | nlTVp.p2ZJ@OIOV.com | 6tSWE.W6sz@whnk.my | TIbih.4r34@xQKA.org | 0M
REl.LqBq@HneI.org | OgUl8.tDq5@nXuM.com | CqhQj.IOaO@LBNv.org | AHLK0.xZAo@gMMX.org | 8lFjA.JOnf@OauW.com | dc6RS.lVud@ipVj.my |
C3NB0.sSSh@AddY.org | I5BlL.2GZL@pVqv.my | 37gBV.mxrk@FVho.com | PfeTO.uNSN@CGNU.com | 11bIA.Pj2T@lbOt.com |
```



```
Dequeue the first element of the priority queue line by line:

zi11f.G0Mx@ZClb.org
zRXvo.Hcaq@Nrtp.com
zCH8P.Rjvr@LEOA.my
xkEQs.02ms@eIAD.org
xadky.lF8R@wYLA.com
wVA78.tBQt@MfMI.org
vFJuc.Ty6t@laOq.my
uwVIg.qjNt@bIHJ.com
tALhL.sg7X@lJsQ.org
siimr.8bHB@yIZG.com
```

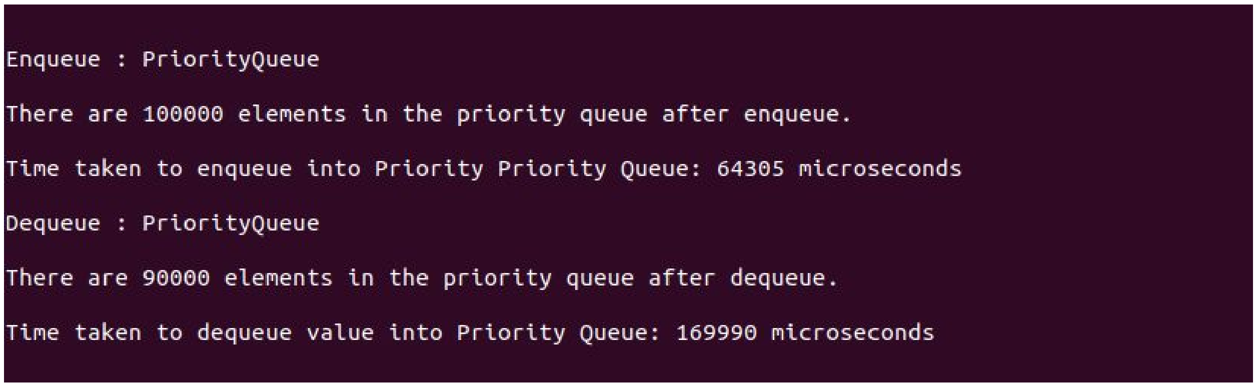


```
PriorityQueue after dequeue:

s3BwB.FUmT@jIkI.org | s1LsO.VGAG@UnZA.org | pbWSk.qGvu@TSsd.org | nyVL4.gksE@tmCO.com | rdiEE.qiRE@HVxI.org | jgE6m.f6L6@aooI.org
 | oOJxK.MO6R@HXPy.com | kA6Hq.lh6z@obDU.my | lQRpI.tAp3@uhnO.com | rXQDT.MgTW@LJXC.com | rTNsb.IdjR@SIRb.org | ecZ36.Yw0K@LjyE.o
rg | c7pBu.0Xnb@itpJ.org | UOUHd.PmJN@DBrG.org | jqkF0.hO41@PfAQ.my | jeEY6.l1xi@EXDW.com | eN2Md.NupH@ZyYa.org | kwKmN.ZlbB@cLRT
.org | QVgWJ.ltyD@lHrI.com | jhzz0.jHZH@UHbK.my | nlTVp.p2ZJ@OIOV.com | hc76K.Q5Oz@ewUR.org | pQnc8.r5GO@kwoc.org | eOg2A.DhdB@hj
hx.my | dc6RS.lVud@ipVj.my | IXYQv.JOKX@MQMW.my | OVztJ.geVC@eMQL.org | Moeyd.DOeE@OvDu.com | SFoaB.BGZB@pspF.org | UlZ7h.AeD1@fg
yS.com | aoGYv.CogI@AGRM.org | gW8k6.lsxt@CnDl.com | AHLK0.xZAo@gMMX.org | cFF4C.Y6D1@toRC.com | PcNGP.xVhc@qrfT.org | cYruY.Psc4
@vkFy.com | fJWmO.yrjQ@bIPh.my | Hp5V6.J0sk@Pylo.org | ApcUo.Rij8@BeFp.org | eKxqL.0OaY@cnPm.org | d01vn.OYvQ@oaIS.com | hkYK5.bv
Ll@TpRY.org | X8UDG.jEm1@RSsH.my | Zwt7p.jX6u@KeJX.my | f7s4S.WCSx@hwsG.my | Me8gy.x2d0@Kxvm.my | m3aFK.LCCe@VHUU.com | XVqK1.3PF
J@OGQl.org | YZcLG.FAJN@YAsi.org | YJuA0.hSlt@AEwb.com | M72ug.At3M@jLHn.my | 7qTAf.HrSb@sQVj.com | 7sXVp.yyHo@YgPU.com | 3B7FT.6
YNL@uxKm.org | CqhQj.IOaO@LBNv.org | 46e0B.XDVk@RNHa.org | JpmaR.s7N4@hJyc.my | 0psVp.RVIL@OQdK.my | FEeTu.rAPF@aUaB.my | LTFMv.2
pmY@ekjx.org | NzuiV.83aj@bXHc.my | 8GnS5.JNxs@MEZj.my | C3izO.AHQ7@jMEc.com | GbJ4r.IFTQ@EOYn.org | QnAGs.PaRu@devJ.org | 0sF0x.
LooP@ivSv.org | 2Hdwf.ZRJY@mrcy.org | KPRel.W8LD@hApl.com | 8lFjA.JOnf@OauW.com | I5BlL.2GZL@pVqv.my | 37gBV.mxrk@FVho.com | CTsM
d.K0Im@uxBD.my | JCOiE.TTqY@qfvv.my | DLAJz.mbiG@Gylc.org | eVdgJ.4vhm@UCFi.com | 85skY.K7q6@LBuQ.my | 11bIA.Pj2T@lbOt.com | 6W5b
4.xTex@BwoJ.my | 0bK1K.i0C4@YKGX.com | 65mBm.ML7H@LMGs.my | YgiAW.fr6E@UELh.org | 8c0QF.HrDY@sZll.com | PfeTO.uNSN@CGNU.com | RSe
AQ.6mIQ@RFon.org | 3zQCV.13FA@Plvk.my | OgUl8.tDq5@nXuM.com | C3NB0.sSSh@AddY.org | 6tSWE.W6sz@whnk.my | TIbih.4r34@xQKA.org | 0M
REl.LqBq@HneI.org |
```



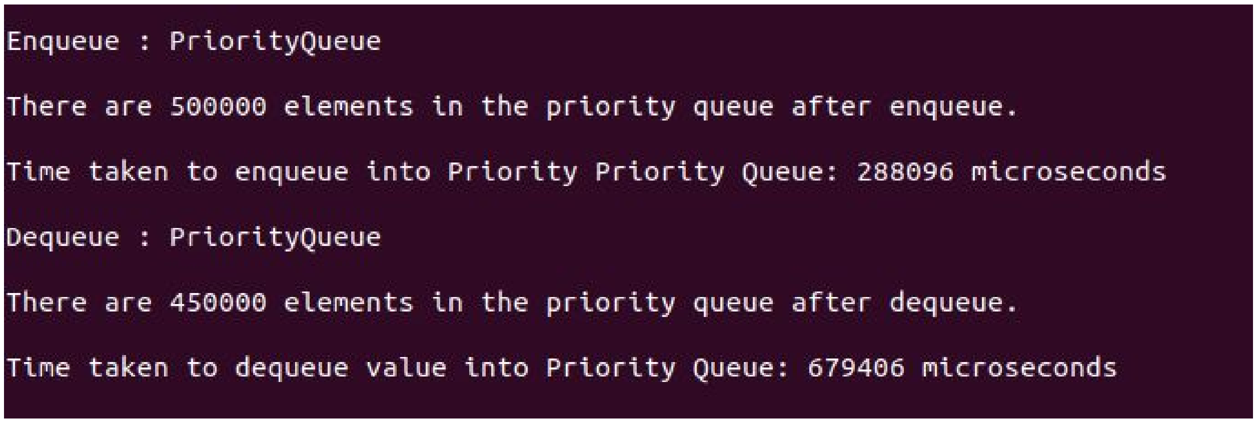
```
Enqueue : PriorityQueue

There are 100 elements in the priority queue after enqueue.

Time taken to enqueue into Priority Priority Queue: 38 microseconds

Dequeue : PriorityQueue

There are 90 elements in the priority queue after dequeue.

Time taken to dequeue value into Priority Queue: 31 microseconds
```

5.3.2 Set B



```
Enqueue : PriorityQueue

There are 100000 elements in the priority queue after enqueue.

Time taken to enqueue into Priority Priority Queue: 64305 microseconds

Dequeue : PriorityQueue

There are 90000 elements in the priority queue after dequeue.

Time taken to dequeue value into Priority Queue: 169990 microseconds
```

5.3.3 Set C



```
Enqueue : PriorityQueue

There are 500000 elements in the priority queue after enqueue.

Time taken to enqueue into Priority Priority Queue: 288096 microseconds

Dequeue : PriorityQueue

There are 450000 elements in the priority queue after dequeue.

Time taken to dequeue value into Priority Queue: 679406 microseconds
```

5.4 Summary

In this section, the team will do a short comparison for all 4 algorithms that the team have worked on in this assignment. They are the Hash Table with Chaining, Hash Table with Linear Probing, AVL Tree and Priority Queue with Heap. In this comparison, we will only compare the largest dataset, which is 500k lines of email (Set C) because it is easier to see differences.

From the results above, Hash Table with Chaining requires 0.759s for the insertion and 0.001s for searching. On the other hand, Hash Table with Linear Probing requires 299s for the insertion and 0 search for searching. Moreover, AVL Tree requires 2.19s for the insertion and 0.002s for searching. Last but not least, Priority Queue with Heap requires 0.288s for enqueue and 0.679s for dequeue. Thus, we can conclude that Hash Table with Linear Probing is the slowest while Priority Queue with Heap is the fastest in terms of insertion among the 4 algorithms .

6.0 Conclusion

It was a wonderful learning experience for the team while working on this assignment. Through the effort of a few weeks, the team has successfully completed and created a hash table algorithm with chaining and linear probing, AVL binary search tree and priority queue using heap.

In the process of writing and understanding the algorithms, the team has faced many challenges. Challenges such as stackoverflow problems, memory leak issues and program crash issues. With extensive research and with the help of lecture slides and lab exercises, the team managed to fix the issues and get a good time complexity for each algorithm

Last but not least, the team would also like to thank Dr Yeoh for the guidance and consultation. Without the guidance of Dr Yeoh, understanding the algorithms would have opposed a bigger challenge.