```
                #####   #####   #####
#####    #   # #     # #     # #     #
#    #    # #  #     #       #       #
#####      #    ######  #####   #####
#    #     #         #       # #
#    #     #   #     # #     # #
#####      #    #####   #####  #######
```

```
/********************************
*Filename:hash.cpp              *
*Login:by932                    *
*AssignmentNo:ass4              *
*DateLastModified:19/10/2014    *
********************************/
#include <iostream>
#include <string.h>
#include <stdlib.h>
#include "hash.h"
using namespace std;

HT::HT()
{
        hashtable[SIZE] = NULL;

        for (int index = 0; index < SIZE; index++)
                hashtable[index] = new htnode();
}

HT::~HT()
{
        for (int i =0; i < 100; i++)
                delete  hashtable[i];

}
void HT::calculate(char input[])
{
        int sum = 0;
        int which = 0;
        for (int i = 0; i < 20; i++)
        {
                sum += static_cast<int>(input[i]);
        }

        which = sum % SIZE;
        hashtable[which]->data.load(input);

}

void HT::findtop()
{
        for (int i = 0; i < SIZE; i++)
        {
                topten(hashtable[i]->data);
        }
        sort(sortarr);
}
bool HT::topten(LIST& pass)
{
        NPtr head;
        //head = hashtable->data.returnhead();
        if (head == NULL)
        {
                return false;
        }
        NPtr temp = head;
        int move = 0;                           // do not change the head pointer
        while (temp != NULL)
        {

                while (sortarr[move].content != ’\0’ && move < 10)
                {
                        move++;
                }
                if (sortarr[move].content == ’\0’)
                {
                        strcpy(sortarr[move].content, temp->content);
                        sortarr[move].count = temp->count;
                        move++;
                }
                else
                {
                        for (int i = 0; i < 10; i++)
                        {
                                if (sortarr[i].count < temp->count)
                                {
                                        strcpy(sortarr[i].content, temp->content);
                                        sortarr[i].count = temp->count;
                                }
                        }
                }
                temp = temp->next;

        }
        return true;

}

void HT::sort(sortnode right[])
{
        for (int i = 9; i > 0; i--)
        {
                for (int q = 0; q < 10; q++)// print out sorted collection
                {
                    cout << right[q].count << " ";
                }
                cout << endl;

                for (int j = 0; j < i; j++)             // perform sorting
                {
                        if (right[j].count > right[j+1].count)
                        {
                                int temp = right[j+1].count;
                                char tempch[21] = {’\0’};
                                strcpy(tempch, right[j + 1].content);

                                right[j+1].count = right[j].count;
                                strcpy(right[j + 1].content, right[j].content);

                                right[j].count = temp;
                                strcpy(right[j].content, tempch);
                }

            }
        }
    }
```

Oct 25 11:22 2014 list.cpp Page 1

```
/********************************
*Filename:list.cpp              *
*Login:by932                    *
*AssignmentNo:ass4              *
*DateLastModified:19/10/2014    *
********************************/
#include <iostream>
#include <cstring>
#include "list.h"
using namespace std;

LIST::LIST()
{
        nonodes = 0;
        head = NULL;

}

LIST::~LIST()
{
        node* temp = head;
        while(temp && temp->next) {
                temp=temp->next;
                delete temp;
        }

        nonodes = 0;
}

bool LIST::load(char input[])
{
        if (head == NULL)       // linked list is empty - so this will be the head node
        {

                NPtr newnode = new node;

                if (newnode == NULL)            // could not allocate memory
                {
                        cout << "Allocation error occured" << endl;
                        return false;
                }
                strcpy(newnode->content, input);
                newnode->count = 1;
                nonodes = 1;
                newnode->next = NULL;
                head = newnode;

        }
        else            // if not here then the linked list exists
        {
                NPtr check = head;
                while (check != NULL)
                {
                        if (strcmp(check->content, input) == 0)
                        {
                                check->count++;
                                break;
                        }
                        if (check->next == NULL)
                        {
                                NPtr newnode = new node;
                                if (newnode == NULL)
                                {
                                        cout << "Allocation error occured" << endl;
                                        return false;
                                }
                                strcpy(newnode->content, input);
                                newnode->count++;
                                nonodes += 1;
                                newnode->next = NULL;
                                check->next = newnode;
                                break;
                        }
                        check = check->next;

                }
        }
        return true;

}

bool LIST::print(ostream&)
{
        if (head == NULL)
        {
                return false;
        }
        NPtr temp = head;                               // do not change the head pointer
        while (temp != NULL)
        {
                cout << temp->content << '\t';
                cout << temp->count;
                cout << endl;
                temp = temp->next;
        }
        return true;

}

NPtr LIST::returnhead()
{
        return head;
}
```

```
Oct 25 11:22 2014  main.cpp Page 1


/********************************
*Filename:main.cpp              *
*Login:by932                    *
*AssignmentNo:ass4              *
*DateLastModified:19/10/2014    *
********************************/
#include <iostream>
#include <fstream>
#include <cstring>
#include "hash.h"
using namespace std;

int main()
{
        int file_len = 0;
        char source_file[20]= {’\0’};
        char data[21] = {’\0’};
        HT start;

        cout << "Enter file name? ";
        cin >> source_file;

        file_len = strlen(source_file);
        ifstream read;
        read.open(source_file);

        if(!read)
        {
                cout << "Cann’t find file"<< endl;
                return 0;
        }
        else if (strcmp(source_file + file_len - 4, ".fna") == 0)
        {
                char del[1025] = {’\0’};

                cout << "This file is a fna file" << endl;
                read.getline (del,1024,’\n’);
                while(!read.eof())
                {
                        for (int i = 0; i < 21; i++)
                                data[i] = ’\0’;

                        for (int i = 0; i < 8; i++)
                        {
                                data[i] = read.get();
                                while (data[i] == ’ ’ || data[i] == ’\n’)
                                {
                                        data[i] = read.get();
                                }

                        }
                        if (!read.good())
                        {
                                read.ignore();
                                read.clear();
                                break;
                        }
                        start.calculate(data);
                }
        }
        else if (strcmp(source_file + file_len - 4, ".txt") == 0)
        {
                cout << "This file is a txt file" << endl;

                while(!read.eof())
                {
                        for (int i = 0; i < 21; i++)
                                data[i] = '\0';

                        for (int i = 0; i < 21; i++)
                        {
                                data[i] = read.get();

                                if (data[i] == ' ' || data[i] == '\n')
                                {
                                        data[i] = '\0';
                                        break;
                                }

                        }
                        if (!read.good())
                        {
                                read.ignore();
                                read.clear();
                                break;
                        }
                        start.calculate(data);
                }
        }
        else
        {
                cout << "This file is not a txt file OR fna file" << endl;
                return 0;
        }
        start.findtop();

        read.close();
        return 0;
}

```

```
Oct 25 11:22 2014  constants.h Page 1


/********************************
*Filename:constants.h           *
*Login:by932                    *
*AssignmentNo:ass4              *
*DateLastModified:19/10/2014    *
********************************/


const int SIZE = 100;   //how bigger hash table
struct sortnode
{
        char content[21];
        int count;
};

```

Oct 25 11:22 2014 hash.h Page 1

```
/********************************
*Filename:hash.h                *
*Login:by932                    *
*AssignmentNo:ass4              *
*DateLastModified:19/10/2014    *
********************************/
#include <iostream>
#include <cstring>
#include "list.h"
using namespace std;

struct htnode
{
   LIST data;
};

typedef htnode* HDNPtr;

class HT
{
        public:
                HT();
                ~HT();
                void calculate(char input[]);
                void findtop();
                bool topten(LIST&);
                void sort(sortnode[]);
        private:
                HDNPtr hashtable[SIZE];
                sortnode sortarr[10];
};

```

Oct 25 11:22 2014 list.h Page 1

```
/********************************
*Filename:list.h                *
*Login:by932                    *
*AssignmentNo:ass4              *
*DateLastModified:19/10/2014    *
********************************/
#include <iostream>
#include "constants.h"
#include <cstring>
using namespace std;
struct node
{
        char content[21];
        int count;
        node *next;
};
typedef node* NPtr;
class LIST
{
        public:
                LIST();
                LIST(const LIST&);
                ~LIST();
                bool load(char input[]);
                bool print(ostream&);
                NPtr returnhead();
        private:
                int nonodes;    // no of nodes in LIST
                NPtr head;    // pointer to list of chars
                NPtr next;
};
```

```
Thanks for submitting constants.h
Thanks for submitting list.h
Thanks for submitting list.cpp
Thanks for submitting hash.h
Thanks for submitting hash.cpp
Thanks for submitting main.cpp

Compiling main.cpp hash.cpp list.cpp

g++ main.cpp hash.cpp list.cpp
```