B1

two desirable properties of a hash function are

1. It should be very fast to compute.
2. It should minimize duplication of output values (collisions).

B2

a

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 12 | 44 | 13 | 88 | 23 | 94 | 11 | 39 | 20 | 16 | 5 |
| h(i) = (2i+5) mod 11 | 7 | 5 | 9 | 5 | 7 | 6 | 5 | 6 | 1 | 4 | 4 |
|  |  |  |  |  |  |  |  |  |  |  |  |
|  | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|  |  | 20 |  |  | 16 | 44 | 94 | 12 |  | 13 |  |
|  |  |  |  |  | 5 | 88 | 39 |  |  |  |  |
|  |  |  |  |  |  | 11 |  |  |  |  |  |

b

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 12 | 44 | 13 | 88 | 23 | 94 | 11 | 39 | 20 | 16 | 5 |
| h(i) = (2i+5) mod 11 | 7 | 5 | 9 | 5 | 7 | 6 | 5 | 6 | 1 | 4 | 4 |
|  |  |  |  |  |  |  |  |  |  |  |  |
|  | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|  | 11 | 20 | 39 | 5 | 16 | 44 | 94 | 12 | 88 | 13 | 23 |

B3

a

b



B4

#include <iostream>

#include <string>

#include <map>

#include <cstdio>

using namespace std;

class *Student* {

private:

public:

*string* name;

*string* Class;

    Student(*string* *name1*,*string* *Class1*) {

        name = *name1*;

        Class = *Class1*;

    }

};

class *studentManager* {

public:

    map<int,*Student*\*> stuList;

    void insert(int *id*,*Student*\* *new\_stu*) {

        stuList[*id*] = *new\_stu*;

    }

    void Delete(int *id*) {

        auto it = stuList.find(*id*);

        if(it != stuList.end()) {

            stuList.erase(it);

        }

    }

*string* infor(int *id*) {

        auto it = stuList.find(*id*);

        if(it != stuList.end()) {

            return it->second->name + "," + it->second->Class;

        } else {

            return "NA,NA";

        }

    }

};

int main() {

*studentManager* manager;

*string* operation;

    while (getline(cin, operation)) {

        if(operation.find("Insert") == 0) {

            int id;

*string* name,Class;

            sscanf(operation.c\_str(), "Insert(%d,%[^,],%[^)])", &id, name.c\_str(), Class.c\_str());

*Student*\* new\_stu = **new** *Student*(name.c\_str(),Class.c\_str());

            manager.insert(id,new\_stu);

        } else if(operation.find("Infor") == 0)  {

            int id;

            sscanf(operation.c\_str(), "Infor(%d)", &id);

            cout<<manager.infor(id)<<endl;

        } else if (operation.find("Delete") == 0) {

            int id;

            sscanf(operation.c\_str(), "Delete(%d)", &id);

            manager.Delete(id);

        }

    }

    return 0;

}