Name: Ayaan Amir

Roll No: 24F-0767

Q1:

```
#include <iostream>

using namespace std;


class Rehash {

   int* t;

   int cap;

   int cnt;

   int emp = -11;

   int rem = -22;


public:

   Rehash(int c = 10) {

      cap = c;

      cnt = 0;

      t = new int[cap];

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

         t[i] = emp;

   }


   int h(int k) { return k % cap; }


   void grow() {

      int oc = cap;

      int* ot = t;
```

```
    cap *= 2;
    t = new int[cap];

    for (int i = 0; i < cap; i++)
        t[i] = emp;

    cnt = 0;

    for (int i = 0; i < oc; i++)
        if (ot[i] >= 0)
            insert(ot[i]);

    delete[] ot;
}

void insert(int k) {
    if ((cnt * 100) / cap > 70)
        grow();

    int idx = h(k);
    while (t[idx] >= 0)
        idx = (idx + 1) % cap;

    t[idx] = k;
    cnt++;
}

bool search(int k) {
    int idx = h(k);
    int st = idx;
```

```
        while (t[idx] != emp) {
            if (t[idx] == k)
                return true;

            idx = (idx + 1) % cap;
            if (idx == st) break;
        }
        return false;
    }

    void removeKey(int k) {
        int idx = h(k);
        int st = idx;

        while (t[idx] != emp) {
            if (t[idx] == k) {
                t[idx] = rem;
                cnt--;
                return;
            }

            idx = (idx + 1) % cap;
            if (idx == st) break;
        }
    }
};

class DoubleHash {
    int* t;
```

```
    int cap;
    int emp = -11;
    int rem = -22;

public:
    DoubleHash(int c = 10) {
        cap = c;
        t = new int[cap];
        for (int i = 0; i < cap; i++)
            t[i] = emp;
    }

    int h1(int k) { return k % cap; }
    int h2(int k) { return 5 - (k % 5); }

    void insert(int k) {
        int idx = h1(k);
        int stp = h2(k);

        while (t[idx] >= 0)
            idx = (idx + stp) % cap;

        t[idx] = k;
    }

    bool search(int k) {
        int idx = h1(k);
        int stp = h2(k);
        int st = idx;
```

```
        while (t[idx] != emp) {
            if (t[idx] == k)
                return true;

            idx = (idx + stp) % cap;
            if (idx == st) break;
        }
        return false;
    }

    void removeKey(int k) {
        int idx = h1(k);
        int stp = h2(k);
        int st = idx;

        while (t[idx] != emp) {
            if (t[idx] == k) {
                t[idx] = rem;
                return;
            }

            idx = (idx + stp) % cap;
            if (idx == st) break;
        }
    }
};

class Bucket {
    int** t;
    int cap;
```

```
    int bs;
    int emp = -11;
    int rem = -22;

public:
    Bucket(int c = 10, int b = 3) {
        cap = c;
        bs = b;

        t = new int* [cap];
        for (int i = 0; i < cap; i++) {
            t[i] = new int[bs];
            for (int j = 0; j < bs; j++)
                t[i][j] = emp;
        }
    }

    int h(int k) { return k % cap; }

    void insert(int k) {
        int idx = h(k);

        for (int j = 0; j < bs; j++) {
            if (t[idx][j] == emp) {
                t[idx][j] = k;
                return;
            }
        }
        cout << "Bucket Full\n";
    }
```

```
    bool search(int k) {
        int idx = h(k);

        for (int j = 0; j < bs; j++)
            if (t[idx][j] == k)
                return true;

        return false;
    }

    void removeKey(int k) {
        int idx = h(k);

        for (int j = 0; j < bs; j++) {
            if (t[idx][j] == k) {
                t[idx][j] = rem;
                return;
            }
        }
    }
};

class Chain {
    struct N {
        int v;
        N* nx;
    };

    N** t;
```

```
    int cap;

public:
    Chain(int c = 10) {
        cap = c;
        t = new N * [cap];
        for (int i = 0; i < cap; i++)
            t[i] = nullptr;
    }

    int h(int k) { return k % cap; }

    void insert(int k) {
        int idx = h(k);
        N* n = new N{ k, t[idx] };
        t[idx] = n;
    }

    bool search(int k) {
        int idx = h(k);
        N* cur = t[idx];

        while (cur) {
            if (cur->v == k)
                return true;
            cur = cur->nx;
        }
        return false;
    }
```

```
    void removeKey(int k) {
        int idx = h(k);
        N* cur = t[idx];
        N* pre = nullptr;

        while (cur) {
            if (cur->v == k) {
                if (pre == nullptr)
                    t[idx] = cur->nx;
                else
                    pre->nx = cur->nx;
                delete cur;
                return;
            }

            pre = cur;
            cur = cur->nx;
        }
    }
};

int main() {
    Rehash r;
    DoubleHash d;
    Bucket b;
    Chain c;

    r.insert(3);
    r.insert(21);
    cout << r.search(20) << endl;
```

```
    d.insert(16);
    d.insert(28);
    cout << d.search(99) << endl;

    b.insert(51);
    b.insert(15);
    cout << b.search(15) << endl;

    c.insert(74);
    c.insert(17);
    cout << c.search(17) << endl;

    return 0;
}
```

}