1.

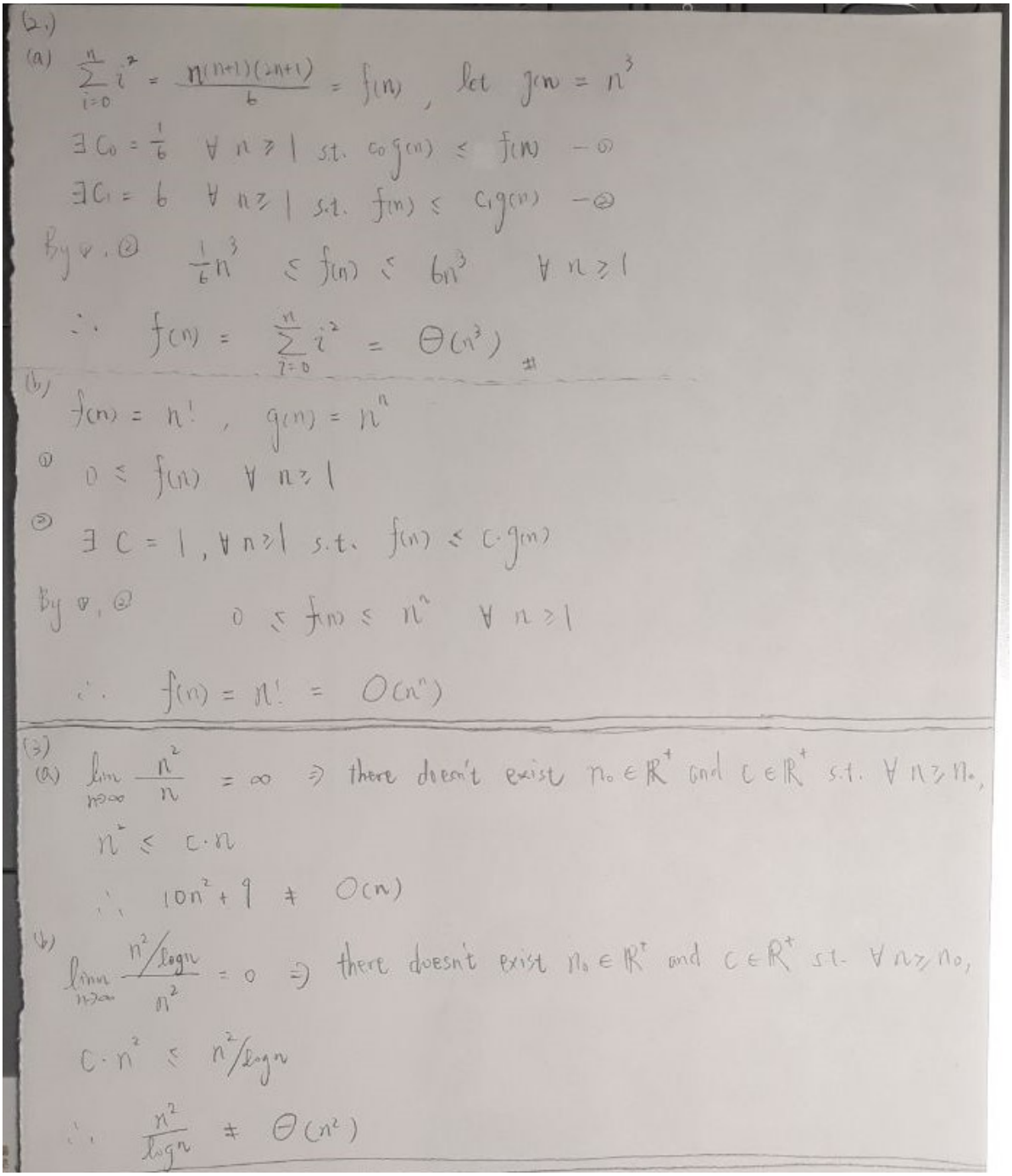
(a)

```
// a[0]: three times
int a[] = {2, 4, 0, -1, 392, 34, 2, 3, 4, 6, 10, 3, 2, 999};
cout << count(a, a + sizeof(a)/sizeof(int), a[0]) << '\n'; // 3
```

(b)

```
using Mat = vector<vector<int> >;
Mat transpose(Mat &m, int Row, int Col) {
    Mat trans(Col, vector<int>(Row));
    for (int i=0; i<Row; ++i)
        for (int j=0; j<Col; ++j)
            trans[j][i] = m[i][j];
    return trans;
}
```

2. && 3.



4.

```cpp
class Complex {
    public:
        int real, img;
        Complex(): real(0), img(0) {};
        Complex(const int &_r, const int &_i): real(_r), img(_i) {};
};
```

5.

```
class Quadratic {
    public:
        int a, b, c;
        Quadratic(const int &_a, const int &_b, const int &_c) {
            a = _a;
            b = _b;
            c = _c;
        }
        Quadratic operator+(const Quadratic &q) {
            return Quadratic(a+q.a, b+q.b, c+q.c);
        }
};
```

6.

```
#ifndef _CSTRING_
#define _CSTRING_
#include <cstring>
#endif

// "Bag" is a "class template"; "Bag<int>" is a "class"
// abstract class
template<typename T>
class Bag {
    public:
        Bag() {}
        virtual ~Bag() = default;

        virtual int Size() { return tail - top; }
        virtual bool IsEmpty() { return tail == top; }
        virtual T Element() const = 0;

        virtual void Push(const T) = 0;
        virtual void Pop() = 0;

    protected:
        T *array; // stores elements
        int capacity; // the maximum number can be used to store elements,
without reallocation
        int top; // array position of top element
        int tail; // array position of tail element
};

template<typename T>
class Queue: public Bag<T> {
    public:
        Queue(int queueCapacity = 10) {
            this->capacity = queueCapacity;
            this->array = new T[queueCapacity];
            this->top = this->tail = size = 0;
        }
        ~Queue() { delete[] this->array; }

        int Size() { return size; }
        bool IsEmpty() { return size == 0; }
        int Capacity() { return this->capacity; }
```

```
41
42          // return "front" element in queue
43          T Element() const {
44              return this->array[this->top % this->capacity];
45          }
46
47          // push an element at the end of queue
48          void Push(const T var) {
49              this->top %= this->capacity;
50              this->tail %= this->capacity;
51
52              // cyclic
53              if (this->tail == this->top) {
54                  // full
55                  if (size == this->capacity) {
56                      this->capacity = (this->capacity + 1) * 2;
57                      T *new_block = new T[this->capacity];
58                      memcpy(new_block, this->array + this->top, sizeof(T)*
(size));
59                      delete this->array;
60
61                      this->array = new_block;
62                      this->tail = size;
63                      this->top = 0;
64                  }
65              }
66              this->array[(this->tail)++] = var;
67              ++size;
68          }
69          void Pop() {
70              if (IsEmpty()) return;
71              this->top %= this->capacity;
72              ++(this->top);
73              --size;
74          }
75      private:
76          int size; // number of elements
77  };
```

7.

$**ABC$

8.

```
1   class Entry {
2       public:
3           int row, col, val;
4   };
5   class SparseMatrix {
6       public:
7           int size;
8           Entry *entries;
9           SparseMatrix(SparseMatrix &_M) {
10              entries = new Entry[_M.size];
11              for (int i=0; i<_M.size; ++i) {
```

```
                entries[i] = _M.entries[i];
            }
        }
};
```

- Computing time (time complexity): $O(N)$ (N is the number of **non-zero** entries.)

9.

```cpp
#include <bits/stdc++.h>
using namespace std;

void insertion_sort(int arr[], int len) {
    for (int i=1; i<len; ++i) {
        int at = arr[i], j;
        for (j=i-1; j>=0; --j) {
            if (arr[j] > at)
                arr[j+1] = arr[j];
            else break;
        }
        arr[j+1] = at;
    }
}

int main() {
    int arr[] = {3, 2, 4, 1, 5};
    insertion_sort(arr, 5);
    for (int i=0; i<5; ++i)
        cout << arr[i] << ' ';
    return 0;
}
```

10.

```cpp
bool is_palin(string &s) {
    int len = s.length();
    for (int i=0; i<len/2; ++i) {
        if (s[i] != s[len-i-1]) return false;
    }
    return true;
}
```