FINAL EXAM A

SECOND SEMESTER OF ACADEMIC YEAR 2023 - 2024

STUDENT ID: __________ NAME: __________ MACHINE NO: __________

Problem	**1**	**2**	**3**	**4**	**5**	**6**	**TOTAL**
	Inheritance& File I/O	**LinkedList**	**String or Sorting**	**Big-O**	**ADT-1& Recursion**	**ADT-2& Recursion**	
Score	**18**	**17**	**15**	**18**	**17**	**15**	**100**

Instructions:

- The time for this exam is ***2.5 hours***.
- Use of anything other than a pencil, pen, notes, the official textbook, and the computer provided by Educational Technology Center is prohibited. In particular, ***no computers or digital devices of any kind you carry with are permitted.***
- Answer submission guides:
  1) Be sure to create an answer folder on the ***D drive.*** The folder is named after ***your student ID + your name***. For example, if your student ID is 2023000123 and your name is "李明", you need to create a folder "***2023000123 李明***". Please be sure to save the corresponding answer (cpp file) to this folder after each problem is completed.
  2) Please check the completed **1.cpp, 2.cpp, 3.cpp, 4.cpp, 5.cpp, 6.cpp** in your folder on the D drive. As soon as the exam time is up, follow the teacher's instructions to submit your folder.

1. (Inheritance & File I/O) (18 pts)

A cake shop uses a file called ***cakes.txt*** to record information about the cakes ordered by the customer. The definition of the ***Cake*** class is known as follows, with ***CubiodCake*** and ***CylinderCake*** being subclasses of the ***Cake*** class. The following program sequentially reads the shape, size (in centimeters), density (in grams/cubic centimeters), and price (in yuan/gram) of each cake from the file ***cakes.txt***. It calculates the price of each cake and then sums it up to obtain the total cost of all cakes. Finally, the total cost is written into the file ***totalcost.txt***.

Please write the missing code according to the requirements (**<u>do not modify</u>** other parts of the code except for areas that require supplementation/modification).

Explanation of data in the ***cakes.txt*** file:

U 12.0 10.0 8.0 1.5 0.02

```
/* U represents CubiodCake, 12.0 10.0 8.0 1.5 0.02 respectively represent:
   length, width, height, density (grams per cubic centimeter), unit price (yuan/gram).
   The unit of length, width, and height is in centimeters
*/
```

Y 10.0 10.0 1.5 0.04

```
/* Y represents CylinderCake, 10.0, 10.0, 1.5, and 0.04 respectively represent:
    height (in centimeters), radius (in centimeters), density (in grams per cubic centimeter),
    unit price (in yuan per gram)
*/
```

```
#include "console.h"
#include <iostream>
#include <fstream>
using namespace std;
class Cake {
public:
    virtual double cakeprice() const = 0;  //纯虚函数
        Cake(double d,double p) {    //构造函数
            density=d;
            unit_price=p;
    };
    double get_unitPrice() const { //成员函数，返回单价：元/克
           return unit_price;
    };
    double get_density() const { //成员函数，返回密度：克/立方厘米
           return density;
    };
private:
    double density;     //蛋糕的密度：克/立方厘米
    double unit_price;  //单价：元/克
};

class CubiodCake : public Cake{
private:
    double length;
    double width;
    double height;
public:
    CubiodCake(double l,double w, double h, double d, double p) : Cake(d,p),length(l), width(w), height(h){}
    double cakeprice() const;
};

class CylinderCake: public Cake {
private:
    double radius;
    double height;
public:
    CylinderCake(double r, double h, double d,double p) :Cake(d,p), radius(r), height(h) {}
    double cakeprice()const;
};
/* (1)请在此处实现 CubiodCake 的成员函数 cakeprice()，返回特定的某个长方体
   形状蛋糕的价格，计算公式：蛋糕价格=体积*密度*单价。*/

/* (2)请在此处实现 CylinderCake 的成员函数 cakeprice()，返回特定的某个圆柱体
    形状蛋糕的价格，计算公式：蛋糕价格=体积*密度*单价。*/
int main() {
     ifstream inputFile("cakes.txt");
  // (3) 请在此处建立 ofstream 类对象 outputFile，并同时打开磁盘文件 totalcost.txt
  // (4) 请修改下面 if 语句的条件，判断是否成功打开了文件
    if (true) {
        cerr << "Error opening files!" << endl;
        return 1;
    }
    double totalCost = 0;
    char type;
    // 以下用 inputFile 对象依次读取 cakes.txt 中的每一行数据，并根据每一行的类型进行计算。
     /* (5) 请修改下面 while 语句中的 true 条件部分，读取 cakes.txt 中每一行数据的蛋糕类型 type，
        提示：请用操作符>>读取，该操作若读取成功返回 ture,若遇文件结束则返回 false。*/
    while (true) {
        if (type == 'U') {
```

```
        double length = 0, width = 0, height = 0, density=0, price=0;
        // (6) 请从磁盘文件依次读入 length，width，height，price
        CubiodCake cubiodCake(length, width, height, density, price);
        totalCost += cubiodCake.cakeprice();

    } else if (type == 'Y') {
            double radius = 0, height = 0,density=0,price=0;
            //(7) 请从磁盘文件依次读入 height，radius，price
            CylinderCake cylinderCake(radius,height,density,price);
            totalCost += cylinderCake.cakeprice();
          }
    }
    cout << totalCost;
    // (8) 请在此处将计算得到的 totalCost 的值写入磁盘文件 totalcost.txt
    // (9) 请在此处关闭打开的所有文件
    return 0;
}
```

2. LinkedList (17pts)

Implement a function:

Node* subList(Node * list, int x, int n);

where ***Node*** is defined as

```
struct Node
 {
    int data;
    Node *next;
 };
```

, ***list*** is pointing to a ***Linked List,*** wherein elements are increasingly ordered, and the function is to find up to ***n*** successive elements starting with the first element greater than or equal to ***x***, building a ***new Linked List*** that contains these elements and returning the pointer of its heading node.

Example:

Assuming that the data in list is {1,3,5,7,9}, *x* is 2, and *n* is 3, then the returned sub list should contain {3,5,7}.

3. String or Sorting(15pts)

In this problem, the program can read a list of students with their scores, sort the students by their scores using ***merge sort***, and then print the sorted list of students. Please implement the functions in the program.

Example:

Input:

{Alice,85}

{Bob,95}

{Charlie,75}

Output:

{Bob,95}

{Alice,85}

{Charlie,75}

4. Big-O (18pts)

Give a tight bound of the nearest runtime complexity class (worst-case) for each of the following code fragments in Big-O notation, in terms of variable N (**the variable N appears in the code** so use that value). As a reminder, when doing Big-O analysis, we write a simple expression that gives only a power of N, such as O(N^2) or O(log N), *not* an exact calculation. It may be helpful to remember this math identity: $\sum_{i=1}^{N} i = \frac{N(N+1)}{2}$.

Four Questions (4.5 pts each)	

#1	int myfunction(Vector<int> vec){
	    int N = vec.size();
	    for (int i = 0; i < N; i++) {
	        for (int j = 0; j < i; j++) {
	            cout << vec[j]<< endl;
	        }
	    }
	}

#2	int myfunction(Vector<int> vec){
	    int N = vec.size();
	    for (int i = 0; i < 10; i++) {
	        for (int j = 0; j < N; j+=N/6) {
	            cout << vec[j]<< endl;
	        }
	    }
	}

#3	int myfunction(Vector <int> &vec, int key) {
	    int N = vec.size();
	    for (int i = 1; i < N; i++) {
	        if (vec[i]==key) {
	            cout << “find ” << key << “ at ” << i;
	            return i;
	        }
	    }
	    return -1;
	}

#4	int myfunction(int N) {
	    if(N <= 1) {
	        // base case
	        return 1;
	    } else {
	        // recursive case
	        return 2*myfunction (N/2);
	    }
	}

Note: this problem does not require programming. Please evaluate the runtime complexity of the ***4*** code fragments in the table above, and modify the definition of ***BigO string array*** in ***4.cpp*** with your solutions.

5. ADT-1&Recursion (17pts)

In this problem, the program can generate all permutations of a given vector of integers using **recursion** and store each permutation in a map with an auto-incremented integer as the key. Please implement the functions in the program.

Example:

Input:

```
Vector<int> nums = {1,  2,  3};
```

Output:

```
Generating permutations of the vector {1, 2, 3} and storing in a map.

All stored permutations are:

  0: 1 2 3

  1: 1 3 2
```

2: 2 1 3

3: 2 3 1

4: 3 2 1

5: 3 1 2

6. ADT-2&Recursion (15pts)

Write a function:

***bool canTransform*(*Vector<int> a, Vector<int> b, int k*);**

in which given a positive integer ***k***, and two vectors ***a*** and ***b*** of the same size with the same set of elements, determine whether it is possible to transform vector ***a*** into vector ***b*** using at most ***k*** swaps.

Here, feasible operation:

swap two elements in the vector.

Example:

Input ***k***, ***a*** and ***b***:

2

{3, 2, 5, 4, 1}

{1, 2, 3, 4, 5}

Output:

true

Explanation:

One swap strategy is {3, 2, 5, 4, 1} --> {3, 2, 1, 4, 5} --> {1, 2, 3, 4, 5}. Therefore, it can be done within 2 swaps.

Note:

You should use a **recursion** to solve the problem.

You are free to write helper functions.

Summary of Relevant Functions and Data Types

We tried to include the most relevant member functions that may be used for the exam, but not all member functions are listed. You are free to use ones not listed here that you know exist.

```
int isalpha(int ch); // Check if the given character is an alphabetic character
int isspace(int c); // Check if the given character is an white space such as Space, '\t', '\n',etc.

class string {
  bool empty() const; // O(1)
  int size() const; // O(1)
  int find(char ch) const; // O(N)
  int find(char ch, int start) const; // O(N)
  string substr(int start) const; // O(N)
  string substr(int start, int length) const; // O(N)
  char& operator[](int index);  // O(1)
  const char& operator[](int index) const; // O(1)
};
string toUpperCase(string str);
string toLowerCase(string str);

class Vector {
  bool isEmpty() const; // O(1)
  int size() const; // O(1)
  void add(const Type& elem); // operator+= used similarly – O(1)
  void insert(int pos, const Type& elem); // O(N)
  void remove(int pos); // O(N)
  Type& operator[](int pos); // O(1)
};

class Grid {
  int numRows() const; // O(1)
  int numCols() const; // O(1)
  bool inBounds(int row, int col) const; // O(1)
  Type get(int row, int col) const; // or operator [][] also works –  O(1)
  void set(int row, int col, const Type& elem); // O(1)
};

class Stack {
  bool isEmpty() const; // O(1)
  void push(const Type& elem); // O(1)
  Type pop(); // O(1)
};

class Queue {
 bool isEmpty() const; // O(1)
 void enqueue(const Type& elem); // O(1)
 Type dequeue(); // O(1)
};

class Map {
 bool isEmpty() const; // O(1)
 int size() const; // O(1)
 void put(const Key& key, const Value& value); // O(logN)
 bool containsKey(const Key& key) const; // O(logN)
 Value get(const Key& key) const; // O(logN)
 Value& operator[](const Key& key); // O(logN)
};
```
Example for Loop: `for (Key key : mymap){…}`

```
class HashMap {
 bool isEmpty() const; // O(1)
 int size() const; // O(1)
 void put(const Key& key, const Value& value); // O(1)
 bool containsKey(const Key& key) const; // O(1)
 Value get(const Key& key) const; // O(1)
 Value& operator[](const Key& key); // O(1)
};
```
Example for Loop: `for (Key key : mymap){…}`

```
class Set {
 bool isEmpty() const; // O(1)
 int size() const; // O(1)
 void add(const Type& elem); // operator+= also adds elements – O(logN)
 bool contains(const Type& elem) const; // O(logN)
 void remove(ValueType value); // O(logN)
};
```
Example for Loop: `for (Type elem : mymap){…}`

```
class Lexicon {
  int size() const; // O(1)
  bool isEmpty() const; // O(1)
  void clear(); // O(N)
  void add(string word); // O(W) where W is word.length()
  bool contains(string word) const; // O(W) where W is word.length()
  bool containsPrefix(string pre) const; // O(W) where W is pre.length()
};
```

Example for Loop: `for (string str : english){…}`