【PROBLEM SET A】 (30%)

✔ This problem is very similar to the ex9-2 problem at lab9.

✔ In this problem, you should implement three classes and the main function. The base class is `Commodity`. And the other classes, `Food` and `Healthy`, are the derived classes.

✔ `Commodity.h`

```
#ifndef COMMODITY_H_
#define COMMODITY_H_

#include <string>
using namespace std;

class Commodity
{
private:

public:
  string name;
  double* score;
  int* price;
  Commodity(string name, int price);
  virtual ~Commodity();
  virtual void cal_score() = 0;
  virtual void show_spec() = 0;
};

#endif //COMMODITY_H_
```

✔ `Food.h`

```
#ifndef FOOD_H_
#define FOOD_H_

#include "Commodity.h"
#include <vector>
using namespace std;

class Food : public Commodity{
private:
  vector<int*> ptr_vec;
  //there are three elements, car, pro and fat, in this vector.
public:
  (constructor/destructor/other function……)
};

#endif //FOOD_H_
```

✔ `Healthy.h`

```
#ifndef HEALTHY_H_
#define HEALTHY_H_

#include "Commodity.h"
```

```
#include <map>
using namespace std;

class Healthy : public Commodity{
private:
  map<string, int> healthy_map;
   //key: ingredient name, value: ingredient value
public:
  (constructor/destructor/other function……)
};

#endif //HEALTHY_H_
```

- ✔ Score:
  - ■ Food: pro_value / price
  - ■ Healthy: summation of all ingredient value / price
- ✔ In the main function, you should read the information from the input file and store it into a vector that contains the `Commodity pointer`. After the input is read, the vector should be sorted by the commodity's name. At last, call the function `show_spec()` in the order of vector. And don't forget to free the memory you use.
- ✔ If you don't know the fast way for sorting a vector, the example 4 in this website is for your information.

  https://shengyu7697.github.io/std-sort/
- ✔ In the input file, one line's information represents one commodity. In each line:
  - ■ The first one is a char and it is 'H' or 'F'.
  - ■ The second one is the commodity's name.
  - ■ The third one is the commodity's price.
  - ■ For a `Healthy`
    - ◆ The fourth one means the ingredient's number.
    - ◆ The rest of line are many pairs of ingredient's name and value.
    - ◆ Be careful, the ingredient's name may be repeated at this time. As a result, you may need to add them together.

      `Ex: H H1 20 2 A 10 A 35. => ingredient A has 45`
    - ◆ When calling the function `show_spec()`, the ingredient should be showed by the order of ingredient's name.
  - ■ For a `Food`
    - ◆ The rest of line are values of car, pro, fat.
- ✔ Execute command: `>./A 1.txt`
- ✔ To pass the test, your program cannot contain memory leaks. You can use `valgrind` to test for memory leaks.
- ✔ Following is `1.txt` and output sample

//1.txt

```
H H1 20 3 A 10 B 35 A 55
H H2 10 5 D 10 B 10 C 10 D 10 B 10
F F1 5 1 2 2
F F2 10 3 5 2
```

//output sample

```
==330991== Memcheck, a memory error detector
==330991== Copyright (C) 2002-2022, and GNU GPL'd, by Julian Seward et al.
==330991== Using Valgrind-3.19.0 and LibVEX; rerun with -h for copyright
info
==330991== Command: ./A 1.txt
==330991==
=======================================
name: F1
price: 5
car: 1
pro: 2
fat: 2
score: 0.4
=======================================
name: F2
price: 10
car: 3
pro: 5
fat: 2
score: 0.5
=======================================
name: H1
price: 20
A: 55
B: 35
score: 4.5
=======================================
name: H2
price: 10
B: 10
C: 10
D: 10
score: 3
=======================================
==330991==
==330991== HEAP SUMMARY:
==330991==     in use at exit: 0 bytes in 0 blocks
==330991==   total heap usage: 58 allocs, 58 frees, 84,329 bytes allocated
==330991==
==330991== All heap blocks were freed -- no leaks are possible
==330991==
==330991== For lists of detected and suppressed errors, rerun with: -s
==330991== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
```

【PROBLEM SET B】 (30%)

Queues are a type of container adaptor, specifically designed to operate in a FIFO context (first-in first-out), where elements are inserted into one end of the container and extracted from the other.

queues are implemented as containers adaptors, which are classes that use an encapsulated object of a specific container class as its underlying container, providing a specific set of member functions to access its elements. Elements are pushed into the "back" of the specific container and popped from its "front".

The underlying container may be one of the standard container class template or some other specifically designed container class. This underlying container shall support at least the following operations: **push, pop, front, back, size, empty**.

In this problem set, you have to implement class **queue**. By template, your class **queue** has to be able to deal with **int** and class **Complex**.

- ✓ **push**: add the element at the end of the queue.
- ✓ **pop**: delete the first element of the queue.
- ✓ **front**: return the value of the first element of the queue.
- ✓ **back**: return the value of the last element of the queue.
- ✓ **size**: return the size of the queue.
- ✓ **empty**: return whether the queue is empty, return true if the queue is empty otherwise return false.

✔ The main program is in **B.cpp**. **DO NOT MODIFY THE MAIN PROGRAM.**

✔ The main function is shown below.

```
// B.cpp
int main(){
    Queue<int> q1;
    if(q1.empty()) cout << "Queue is empty!" << endl;
    else cout << "Queue is not empty!" << endl;

    q1.push(5); cout << "Pushed " << q1.back() << endl;
    q1.push(4); cout << "Pushed " << q1.back() << endl;
    q1.push(3); cout << "Pushed " << q1.back() << endl;

    if(q1.empty()) cout << "Queue is empty!" << endl;
    else cout << "Queue is not empty!" << endl;

    int size = q1.size();
    for(int i=0; i<size; i++){
        cout << q1.front() << " ";
        q1.pop();
    } cout << endl << endl;

    Queue<Complex> q2;
    if(q2.empty()) cout << "Queue is empty!" << endl;
```

```
    else cout << "Queue is not empty!" << endl;

    Complex c1; // real = 0, imag = 0
    q2.push(c1); cout << "Pushed " << q2.back() << endl;
    Complex c2(3.9, 7.1);
    q2.push(c2); cout << "Pushed " << q2.back() << endl;
    c1.set_real(5.6); c1.set_imag(9.4);
    q2.push(c1); cout << "Pushed " << q2.back() << endl;
    Complex c3 = c1 + c2;
    q2.push(c3); cout << "Pushed " << q2.back() << endl;
    c3 = c1 - c2;
    q2.push(c3); cout << "Pushed " << q2.back() << endl;
    c3 += c1;
    q2.push(c3); cout << "Pushed " << q2.back() << endl;
    c3 -= c2;
    q2.push(c3); cout << "Pushed " << q2.back() << endl;

    if(q2.empty()) cout << "Queue is empty!" << endl;
    else cout << "Queue is not empty!" << endl;

    size = q2.size();
    for(int i=0; i<size; i++){
        cout << q2.front() << " ";
        q2.pop();
    } cout << endl;

    return 0;
}
```

✔ To pass the test, your program cannot contain memory leaks. You can use the following command to test for memory leaks.
  > valgrind <your_execution_binary>
  ex: valgrind ./B
  Valgrind reference:
  https://web.stanford.edu/class/archive/cs/cs107/cs107.1234/resources/valgrind.html

✔ You can copy **B.cpp, Queue.h,** and **Complex.h** from /home/share/final/B/.

✔ Please write a "**Makefile**" in your directory to compile your code. Your code can be compiled by typing "**make**", and the name of your program should be **B**.

✔ Compile
  make

✔ The execution results are as follows.

```
$ ./B
Queue is empty!
Pushed 5
Pushed 4
Pushed 3
Queue is not empty!
5 4 3

Queue is empty!
Pushed 0+0j
Pushed 3.9+7.1j
Pushed 5.6+9.4j
Pushed 9.5+16.5j
Pushed 1.7+2.3j
Pushed 7.3+11.7j
Pushed 3.4+4.6j
Queue is not empty!
0+0j 3.9+7.1j 5.6+9.4j 9.5+16.5j 1.7+2.3j 7.3+11.7j 3.4+4.6j
```

✔ Please declare class **Complex** in **Complex.h** and define its functionality in **Complex.cpp**.

```
// Complex.h
class Complex{
private:
    double real;
    double imag;
public:
    // please add constructors
    void set_real(double);
    void set_imag(double);
    void operator=(const Complex &);
    Complex operator+(const Complex &);
    Complex operator-(const Complex &);
    void operator+=(const Complex &);
    void operator-=(const Complex &);
    friend std::ostream &operator<<(std::ostream &os, const
Complex);
};
```

✔ Please declare the template class **Queue** and finish its definition in **Queue.h**.

```
// Queue.h
template <typename S>
struct node{
    S data;
    node<S> *link;
};

template <class T>
class Queue
{
private:
    int len;
    node<T> *head;
    node<T> *tail;
public:
    Queue();
    // please add necessary function members
};
```

【PROBLEM SET C】 (30%)

In the second half of the semester, we learn to use templates to simplify repeated declarations of objects. Therefore, for this topic, I hope that you will use the template method to declare objects and complete a simple library borrowing system.

You have been tasked with implementing a library borrowing system using a template class. The template class provides the basic structure and functionality for managing items in the library.

Your task is to create two classes, Book and Reader, based on the provided template. The Book class should represent a book in the library, while the Reader class should represent a library patron.

Requirements:

The Book class should inherit from the template class, providing the necessary implementation specific to books.

The Reader class should also inherit from the template class, with additional functionality specific to readers.

Each book should have a unique identifier (ID) and a title.

The Reader class should have a unique identifier (ID) and a name.

The library borrowing system should support the following operations:

- Adding a book to the library.
- Registering a new reader.
- Borrowing a book by a reader.
- Returning a borrowed book.
- Displaying the information of books and readers.

Your implementation should demonstrate the proper usage of templates, inheritance, and object-oriented programming principles.

Write the Book and Reader classes based on the provided template class, and implement a test program that showcases the functionality of the library borrowing system. Ensure that the program allows users to perform the necessary operations mentioned above.

✔ The template class is in `Item.h`. **DO NOT MODIFY THE PROGRAMS.**

✔ You can copy all the files you may need from /home/share/final/C/.

ITEM

Template class is defined in **Item.h.**

The .h file declares a template class called Item, where <class T> indicates that it is a template class that can accept different types as template parameters.

Within this class, there are two private member variables, id and name, which are used to store the identification and name of the item. These member variables can only be accessed within the class.

Please use this template class to implement the Book and Reader classes.

```
template <class T>
class Item {
private:
    int id;
    string name;

public:
    Item(int id, std::string name);
    int getId();
    string getName();
    virtual void printInfo() = 0;
};
```

BOOK

The book's ID starts from 1001 and inherits the ID and name from the Item class. Additionally, the book class needs to keep track of its own borrowing status.

USER

The reader's ID starts from 1 and inherits the ID and name from the Item class. A reader can borrow more than one book, but they cannot borrow a book that is already borrowed by another reader.

MAIN

In the main function, the program will continuously prompt the user to enter a user ID. For ease of implementation, if the user enters an ID of -1, the program will request the ID and name for registration. If the user enters an ID of 0, the program will display all book and reader data.

For the book, in addition to its own ID and name, it needs to display its borrowing status. As for the reader, it needs to list all the IDs of the books it has borrowed.

When the user enters a valid reader ID, the program will prompt for a book ID. Based on the status of the book, the program will perform the following actions:

1. If the book is not borrowed by anyone, the program will successfully borrow the book.
2. If the reader has already borrowed the book, the program will return it.
3. If someone else is currently borrowing the book, the program will display the corresponding message.

Please implement the corresponding program logic based on the requirements mentioned above.

※ No need fool-proofing, such as non-existent id or duplicate id.

```
$ ./C
User login: -1
id and name to add Book or Reader
1001 ICP
User login: -1
id and name to add Book or Reader
1002 OOP
User login: -1
id and name to add Book or Reader
1003 Algorithm
User login: -1
id and name to add Book or Reader
1 Lowry
User login: -1
id and name to add Book or Reader
2 Boki
User login: 1
Book id: 1001
User login: 2
Book id: 1001
This book is being borrowed
User login: 2
Book id: 1002
User login: 0
-----------------------------------
Book ID: 1001
Book Name: ICP
Is Borrowed: Yes
Book ID: 1002
Book Name: OOP
Is Borrowed: Yes
Book ID: 1003
```

```
Book Name: Algorithm
Is Borrowed: No
Reader ID: 1
Reader Name: Lowry
Borrowed Items:
- 1001
Reader ID: 2
Reader Name: Boki
Borrowed Items:
- 1002
----------------------------------
User login: 1
Book id: 1001
Return the book
User login: 2
Book id: 1001
User login: 2
Book id: 1003
User login: 0
----------------------------------
Book ID: 1001
Book Name: ICP
Is Borrowed: Yes
Book ID: 1002
Book Name: OOP
Is Borrowed: Yes
Book ID: 1003
Book Name: Algorithm
Is Borrowed: Yes
Reader ID: 1
Reader Name: Lowry
Borrowed Items:
Reader ID: 2
Reader Name: Boki
Borrowed Items:
- 1002
- 1001
- 1003
----------------------------------
User login: ^C
```

【PROBLEM SET D】 (40%)

The C++ Standard Template Library (STL) is a powerful set of template classes and functions provided by the C++ Standard Library. It offers a wide range of data structures and algorithms that can be used to simplify and accelerate the development of C++ programs. By leveraging the STL, you can write more concise and efficient code while focusing on the problem at hand rather than low-level implementation details.

The STL consists of several components:

- Container
- Algorithm
- Iterator
- Function Object
- Utility

In this problem set, you must implement a simple Container with list and vector using template and diamond inheritance.

✔ The main programs are in `D-1.cpp`, `D-2.cpp`, `D-3.cpp`. **DO NOT MODIFY THE MAIN PROGRAMS.**

✔ You can modify and add proper prototypes to all the header files.

✔ To pass the test, your program cannot contain memory leaks. You can use the following command to test for memory leaks.

```
> valgrind <your_execution_binary>
ex: valgrind ./D-1
```

Valgrind reference:
https://web.stanford.edu/class/archive/cs/cs107/cs107.1234/resources/valgrind.html

✔ You can copy `D-1.cpp`, `D-2.cpp`, `D-3.cpp`, `Node.h`, `Makefile`, `D-1`, `D-2`, and `D-3` from /home/share/final/D/.

✔ Demo command:

Level 1 (20%):

```
make D-1
./D-1
```

Level 2 (10%):

```
make D-2
./D-2
```

Level 3 (10%):

```
make D-3
./D-3
```

Makefile usage:

```
make: it will compile all the problems
make clean: it will remove all the executable file
```

✔ You can execute `D-1`, `D-2`, and `D-3` to see what the result looks like.

PREPROCESSING COMPLETE THE ABSTRACT CLASS (0 points)

Please complete all constructors, destructors, and the declaration of required member functions which vector and list have such as insert, erase, size, empty, clear, begin, end, and display in AbstractContainer.hpp.

NOTE

In this problem set, you only need to write the declaration and the definition in same file named *.hpp because the template doesn't allow you to write them separately.

```
// AbstractContainer.hpp

template <class T>
class AbstractContainer
{
protected:
    T *arr;
    int len;

public:

    // all constructors and destructor

    // pure virtual function of insert, erase, size, empty,
clear, begin, end, and display

};

//member definition
```

D-1 COMPLETE THE LIST (20 points)

AbstractContainer.hpp and List.hpp must be completed in order for D-1.cpp to run correctly.

List.hpp:

Please complete all constructors, destructors, and the required member functions which List needs with the following requirements. For D-1, you only have to maintain the characteristic of List in diamond inheritance architecture.

(1) Constructor:

(a) Call the AbstractContainer class constructor and build its own double-linked list structure.

(2) Destructor:

(a) Delete all memory if necessary.

(3) insert:
(a) It needs two arguments, the first one is position and the second one is inserted element value.

(4) erase:
(a) It needs an argument which means which position needs to be erased.

(5) size:
(a) Return the size of double-linked list.

(6) empty:
(a) Return if the double-linked list is empty.

(7) clear:
(a) Clear the double-linked list.

(8) begin:
(a) Return the index of the first one in the double-linked list.

(9) end:
(a) Return the index of the last one in the double-linked list.

(10) Display:
(a) Display the double-linked list in accordance with the example.

(11) head variable:
(a) A pointer points to the first Node of the double-linked list.

(12) tail variable:
(a) A pointer points to the last Node of the double-linked list.

```
// List.hpp
//include necessary header files

template <class T>
class List : virtual public AbstractContainer<T>
{
public:
    // all constructors and destructor

    // override the pure virtual function

protected:
    Node<T> *head;
    Node<T> *tail;
};

//member definition
```

The sample output of level 1:

```
>make D-1
>./D-1
Original:
        nullptr <-> -1 <-> -1 <-> -1 <-> -1 <-> -1 <-> 3 <-> 5 <->
8 <-> 1 <-> 6 <-> nullptr
After insert(2, 11), insert(6, -5), and insert(11, 20):
        nullptr <-> -1 <-> -1 <-> 11 <-> -1 <-> -1 <-> -1 <-> -5
<-> 3 <-> 5 <-> 8 <-> 1 <-> 20 <-> 6 <-> nullptr
After erase(0):
        nullptr <-> -1 <-> 11 <-> -1 <-> -1 <-> -1 <-> -5 <-> 3 <-
> 5 <-> 8 <-> 1 <-> 20 <-> 6 <-> nullptr
List size: 12
After erase(11):
        nullptr <-> -1 <-> 11 <-> -1 <-> -1 <-> -1 <-> -5 <-> 3 <-
> 5 <-> 8 <-> 1 <-> 20 <-> nullptr
After erase(5):
        nullptr <-> -1 <-> 11 <-> -1 <-> -1 <-> -1 <-> 3 <-> 5 <->
8 <-> 1 <-> 20 <-> nullptr
List size: 10
times: 5
nullptr <-> 11 <-> -1 <-> -1 <-> -1 <-> 3 <-> 5 <-> 8 <-> 1 <->
nullptr
nullptr <-> -1 <-> -1 <-> -1 <-> 3 <-> 5 <-> 8 <-> nullptr
nullptr <-> -1 <-> -1 <-> 3 <-> 5 <-> nullptr
nullptr <-> -1 <-> 3 <-> nullptr
nullptr <-> nullptr
List size: 0
>
```

D-2 COMPLETE THE VECTOR (10 points)

AbstractContainer.hpp and Vector.hpp must be completed in order for D-2.cpp to run correctly.

Vector.hpp:

Please complete all constructors, destructors, and the required member functions which Vector needs with the following requirements. For D-2, you only have to maintain the characteristic of Vector in diamond inheritance architecture.

(1) Constructor:
- (a) Call the AbstractContainer class constructor and build its dynamic array structure following the STL's vector rules.
- (b) Capacity will double when there is no empty capacity for adding a new item.

(2) Destructor:
- (a) Delete all memory if necessary.

(3) insert:
- (a) It needs two arguments, the first one is position and the second one is inserted element value.

(4) erase:
- (a) It needs an argument which means which position needs to be erased.

(5) size:
- (a) Return the size in use.

(6) empty:
- (a) Return if the dynamic array is empty.

(7) clear:
- (a) Just clear the dynamic array, not release the memory.

(8) begin:
- (a) Return the start variable.

(9) end:
- (a) Return the finish variable.

(10) display:
- (a) Display the dynamic array in accordance with the example.

(11) push_back:
- (a) Consider it as inserting from the end.

(12) pop_back:
- (a) Consider it as erasing from the end.

(13) capacity:
- (a) return the total capacity.

(14) [] operator:
- (a) Implement the [] of Vector.

(15) data variable:

(a) A pointer points to the first address of the dynamic array.

(16) start variable:

(a) Record the first index of the dynamic array in use.

(17) finish variable:

(a) Record the last index of the dynamic array in use.

(18) end_of_storage variable:

(a) Record the number of the dynamic array that can use.

```
// Vector.hpp
//include necessary header files

template <class T>
class Vector : virtual public AbstractContainer<T>
{
public:
    // all constructors and destructor

    // declare the member function which Vector required

    // override the pure virtual function

    // add the necessary operator

protected:
    T *data;
    int start;
    int finish;
    int end_of_storage;
};

//member definition
```

The sample output of level 2:

```
>make D-2
>./D-2
V_ptr->capacity(): 5
After declaration:
        display(): -1 -1 -1 -1 -1
After insert(4, 4):
        display(): -1 -1 -1 -1 4 -1 * * * *
After push_back(1), push_back(3), insert(7, 6), and insert(8, 9):
        display(): -1 -1 -1 -1 4 -1 1 6 9 3
After push_back(10), pop_back():
        display(): -1 -1 -1 -1 4 -1 1 6 9 3 * * * * * * * * * *
V_ptr->capacity(): 20
After erase(3), erase(5):
        display(): -1 -1 -1 4 -1 6 9 3 * * * * * * * * * * * *
cout: -1 -1 -1 4 -1 6 9 3
After erase(V_ptr->begin()), erase(V_ptr->end()):
        display(): -1 -1 4 -1 6 9 * * * * * * * * * * * * * *
After erase(V_ptr->begin()), erase(V_ptr->end()):
        display(): -1 4 -1 6 * * * * * * * * * * * * * * * *
After erase(V_ptr->begin()), erase(V_ptr->end()):
        display(): 4 -1 * * * * * * * * * * * * * * * * * *
After erase(V_ptr->begin()), erase(V_ptr->end()):
        display(): * * * * * * * * * * * * * * * * * * * *
size(): 0
V_ptr->capacity(): 20
>
```

D-3 DEFINE YOUR STL (10 points)

AbstractContainer.hpp, List.hpp, Vector.hpp, and MySTL.hpp must be completed in order for D-3.cpp to run correctly.

MySTL.hpp:

Please complete all constructors, destructors, and the required member functions which MySTL needs with the following requirements. For D-3, you only have to maintain the characteristic of MySTL in diamond inheritance architecture.

(1) Constructor:
- (a) Call the AbstractContainer class constructor, List class constructor, and Vector class constructor and make it realize the dynamic binding for D-3.cpp in diamond inheritance architecture.

(2) Destructor:
- (a) Delete all memory if necessary.

(3) All other members:
- (a) Using List's class API by default.

```
// MySTL.hpp

//include necessary header files

template <class T>
class MySTL : public List<T>, public Vector<T>
{
public:
    // all constructors and destructor

    // customize the pure virtual function implementation

};

//member definition
```

The sample output of level 3:

```
>make D-3
>./D-3
================================ MySTL ================================
Original:
    nullptr <-> -1 <-> -1 <-> -1 <-> -1 <-> -1 <-> 3 <-> 5 <-> 8 <-> 1 <-> 6 <-> nullptr
After insert(2, 11), insert(6, -5), and insert(11, 20):
    nullptr <-> -1 <-> -1 <-> 11 <-> -1 <-> -1 <-> -1 <-> -5 <-> 3 <-> 5 <-> 8 <-> 1 <-> 20 <->
```

```
6 <-> nullptr
After erase(0):
    nullptr <-> -1 <-> 11 <-> -1 <-> -1 <-> -1 <-> -5 <-> 3 <-> 5 <-> 8 <-> 1 <-> 20 <-> 6 <->
nullptr
List size: 12
After erase(11):
    nullptr <-> -1 <-> 11 <-> -1 <-> -1 <-> -1 <-> -5 <-> 3 <-> 5 <-> 8 <-> 1 <-> 20 <-> nullptr
After erase(5):
    nullptr <-> -1 <-> 11 <-> -1 <-> -1 <-> -1 <-> 3 <-> 5 <-> 8 <-> 1 <-> 20 <-> nullptr
List size: 10
times: 5
nullptr <-> 11 <-> -1 <-> -1 <-> -1 <-> 3 <-> 5 <-> 8 <-> 1 <-> nullptr
nullptr <-> -1 <-> -1 <-> -1 <-> 3 <-> 5 <-> 8 <-> nullptr
nullptr <-> -1 <-> -1 <-> 3 <-> 5 <-> nullptr
nullptr <-> -1 <-> 3 <-> nullptr
nullptr <-> nullptr
List size: 0
================================ Vector ================================
V_ptr->capacity(): 10
After declaration:
    display(): -1 -1 -1 -1 -1 3 5 8 1 6
After pop_back()*5, insert(4, 4):
    display(): -1 -1 -1 -1 4 -1 * * * *
After push_back(1), push_back(3), insert(7, 6), and insert(8, 9):
    display(): -1 -1 -1 -1 4 -1 1 6 9 3
After push_back(10), pop_back():
    display(): -1 -1 -1 -1 4 -1 1 6 9 3 * * * * * * * * * *
V_ptr->capacity(): 20
After erase(3), erase(5):
    display(): -1 -1 -1 4 -1 6 9 3 * * * * * * * * * * * *
cout: -1 -1 -1 4 -1 6 9 3
After erase(V_ptr->begin()), erase(V_ptr->end()):
    display(): -1 -1 4 -1 6 9 * * * * * * * * * * * * * *
After erase(V_ptr->begin()), erase(V_ptr->end()):
    display(): -1 4 -1 6 * * * * * * * * * * * * * * * *
After erase(V_ptr->begin()), erase(V_ptr->end()):
    display(): 4 -1 * * * * * * * * * * * * * * * * * * *
After erase(V_ptr->begin()), erase(V_ptr->end()):
    display(): * * * * * * * * * * * * * * * * * * * * *
size(): 0
V_ptr->capacity(): 20
>
```

【PROBLEM SET E】 (40%)

Minecraft is a sandbox game developed by Mojang Studios. The game was created by Markus "Notch" Persson in the Java programming language. Following several early private testing versions, it was first made public in May 2009 before being fully released in November 2011, with Notch stepping down and Jens "Jeb" Bergensten taking over development. Minecraft is the best-selling video game in history, with over 238 million copies sold and nearly 140 million monthly active players as of 2021, and has been ported to several platforms.

In this problem set, you have to implement a simple 2D Minecraft game with 4 block types, player, and mining.

- ✔ The main programs are in `E-1.cpp, E-2.cpp, E-3.cpp`. **DO NOT MODIFY THE MAIN PROGRAMS.**
- ✔ You can modify and add proper prototypes to all the header files
- ✔ To pass the test, your program cannot contain memory leaks. You can use the following command to test for memory leaks.
  ```
  > valgrind <your_execution_binary>
  ex: valgrind ./E-1
  ```
  Valgrind reference:
  https://web.stanford.edu/class/archive/cs/cs107/cs107.1234/resources/valgrind.html
- ✔ You can copy `E-1.cpp, E-2.cpp, E-3.cpp, Minecraft.h, Minecraft.cpp, Block.h, Player.h, Position.h, makefile, E-1, E-2, E-3` and `map` from /home/share/final/A/.
- ✔ Compile
  Level 1 (20%):
  ```
  make E-1
  ```
  Level 2 (10%):
  ```
  make E-2
  ```
  Level 3 (5%):
  ```
  make E-3
  ```
  Level 4 (5%):
  ```
  make E-3
  ```
- ✔ You can execute `./E-1, ./E-2, ./E-3` to see what the result looks like.

```
// Minecraft.h
enum class ArrowKey { UP, DOWN, LEFT, RIGHT };

class Minecraft {
    int sizeX;
    int sizeY;
```

```
    // 2D array of Block*
    Block**(*map);

    Player* player = nullptr;
    bool exit = false;
    Block* selectedBlock = nullptr;

  public:
    int getSizeX() { return sizeX; };
    int getSizeY() { return sizeY; };
    void run();

    // Level1
    // Call buildMap
    Minecraft(string mapFilename);

    // delete blocks in map
    ~Minecraft();

    // Build map from map file
    void buildMap(string mapFilename);

    // Render blocks and player
    void render();

  private:
    // Level2
    // If 'a', player move left
    // Else if 'd', player move right
    // Else if 'x', exit game
    // Else if 'f', mine the selected block
    // After player move, selectedBlock = nullptr
    void onNormalKeyPress(char key);

    // Level3
    // Selected block
    // If selectedBlock == nullptr
    //   Selected block under player
    // Else
    //   Select the next block in the given direction
    //   of the current selected block
    //   The block also has to be inside the
Player::mineDistance
    void onArrowKeyPress(ArrowKey key);
};
```

```
// Block.h
class Block {
   public:
    static const int sizeY = 4;
    static const int sizeX = 8;
   protected:
    bool selected = false;
    Position pos;
    int hardness;
    int minedCount = 0;

   public:
    void setSelected(bool selected) { this->selected =
selected; }
    Position getPos() { return pos; }
    virtual ~Block() = default;

    // Level1
    // Construct block
    Block(Position pos);

    virtual char** render() = 0;

    // Level3
    // Mine the block
    // Increase minedCount
    // Return true if minedCount >= hardness, else return false
    bool mine();
};
```

```
// Player.h
class Player {
   public:
    const static int sizeX = 8;
    const static int sizeY = 8;
    const static int mineDistance = 5;

    enum class Movement { LEFT, RIGHT };

   private:
    Position pos;
    Block**(*map);
    int mapSizeX;
    int mapSizeY;

   public:
    Position getPos() { return pos; };

    // Level1
```

```
    // Construct player
    Player(Position pos, Block*** map, int mapSizeY, int
mapSizeX);

    // Render player
    char** render();

    // Level2
    // Move the player left or right
    // Player can go up 1 block
    // Player will fall if there is no block below
    // Need to check for x bound
    void move(Movement movement);

    // Level3
    // Mine a block
    // Call block->mine()
    // If the block below player is mined, player will fall
    // Return true if block is mined, else return false
    bool mine(Position pos);
};
```

E-1 BUILD THE MAP(20 points)

Please complete all constructors, destructors, and the required level 1 member functions in each above class declaration with the following requirements. For E-1, you only have to render one frame from the map file. You also have to add class **Dirt**, class **Leaf**, class **Rock**, and class **Trunk** in **Dirt.h, Leaf.h, Rock.h,** and **Trunk.h**, the above classes all inherited from class **Block.**

(1) Minecraft.h

(a) Constructor: Call buildMap.

(b) Destructor: Delete all the blocks.

(c) buildMap: Build the map from mapfile. The first line of the map file indicates the size of the map, and the remaining rows are the block arrangement.

(d) **render:** Render the map, call **system("clear")** first.

(2) Player.h

(a) Constructor: Initialize member variables.

(b) Render: Render the block.

(3) Block.h

(a) **Constructor:** Takes position as the argument.

(4) `Dirt.h`, `Leaf.h`, `Rock.h`, and **`Trunk.h`**

(a) `Constructor`: Takes position as the argument and set the hardness for each type of block.

(b) `Render`: Render the block.

The properties of each block and player are as followed

Dirt	Leaf	Trunk	Rock	Player
hardness = 2	hardness = 1	hardness = 4	hardness = 6	
########	########	||||||||	%&%&%&%&	+----+
@@@@@@@@	########	||||||||	&%&%&%&%	|+__+|
@@@@@@@@	########	||||||||	%&%&%&%&	+----+
@@@@@@@@	########	||||||||	&%&%&%&%	|======|
				| == |
				| |==| |
				| |
				| |

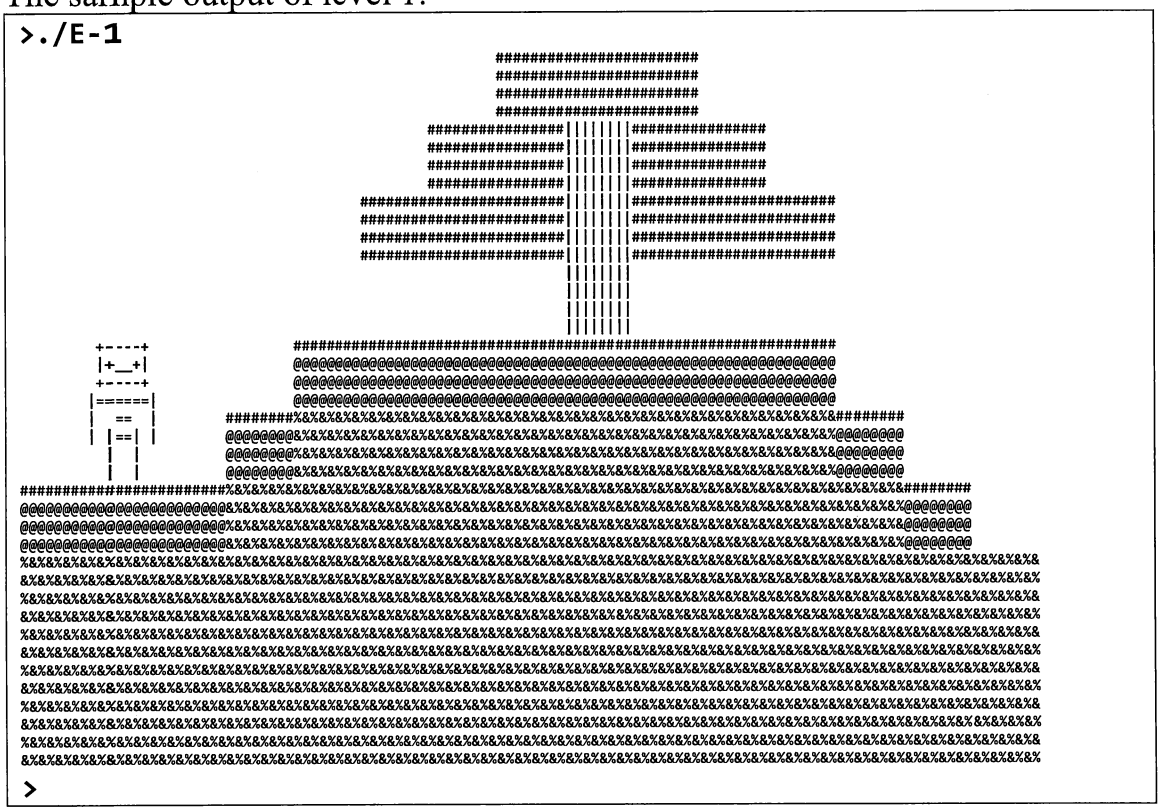
The sample input map file of E-01:

```
10 15
aaaaaaalllaaaaa
aaaaaalltllaaaa
aaaaallltlllaaa
aaaaaaaataaaaaa
aaaaddddddddaaa
asadrrrrrrrrdaa
dddrrrrrrrrrda
rrrrrrrrrrrrrr
rrrrrrrrrrrrrr
rrrrrrrrrrrrrr
```

(a) `10 15` means that **`sizeY = 10`** and **`sizeX = 15.`**

(b) `a` means **`air (no block), d`** means **`Dirt, l`** means **`Leaf, t`** means **`Trunk,`** and `r` means **`Rock.`**

The sample output of level 1:



E-2 PLAYER MOVEMENT (10 points)

Please complete all the required level 2 member functions in each above class declaration with the following requirements. For E-2, you need to implement player movement. The player will move right if `'d'` is pressed and will move left if `'a'` is pressed. The player can move up to the adjacent higher block if the block is only one block tall. Also, if there is no block under the player after moving, the player will fall.

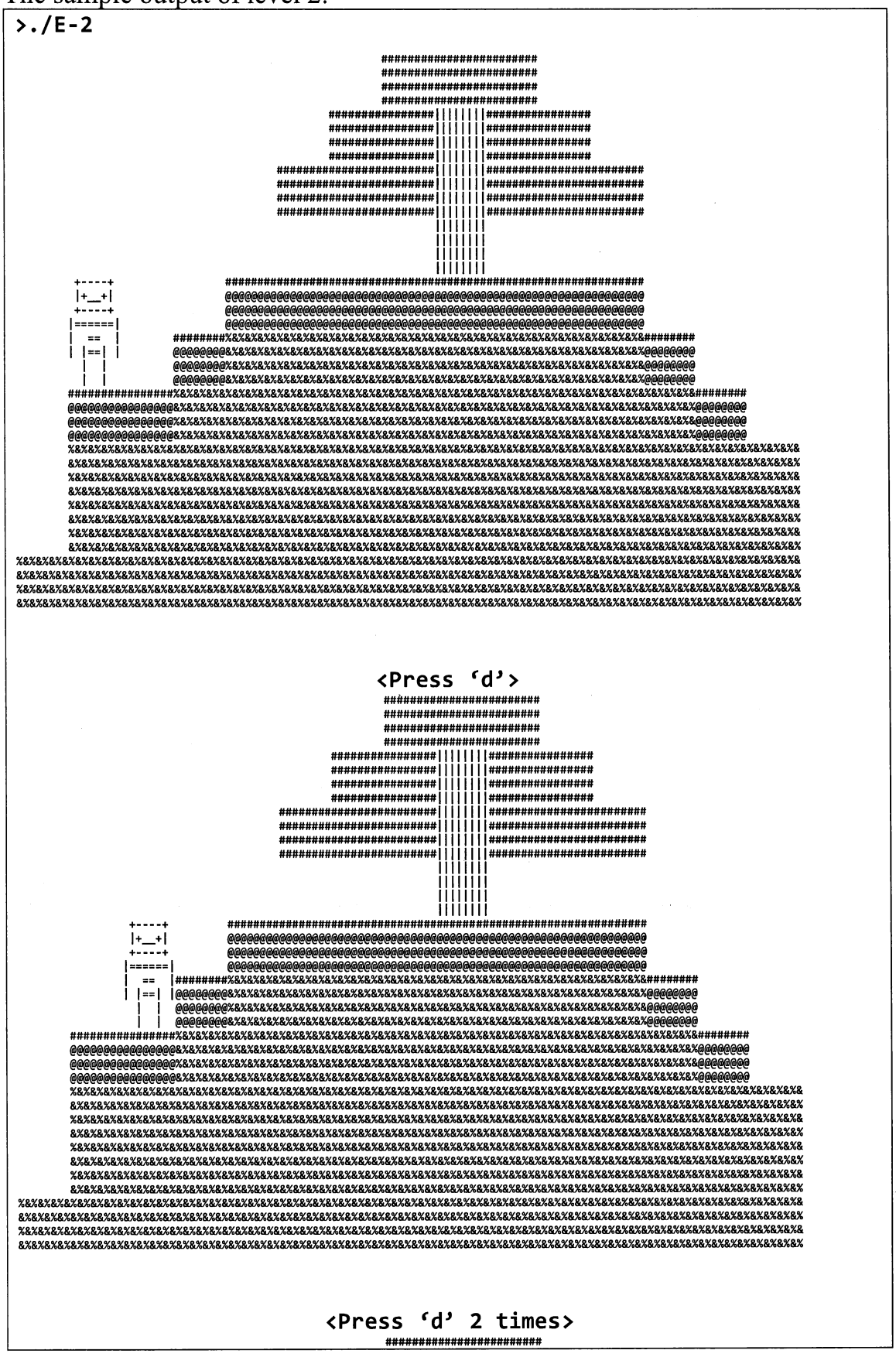
(1) `Minecraft.h`

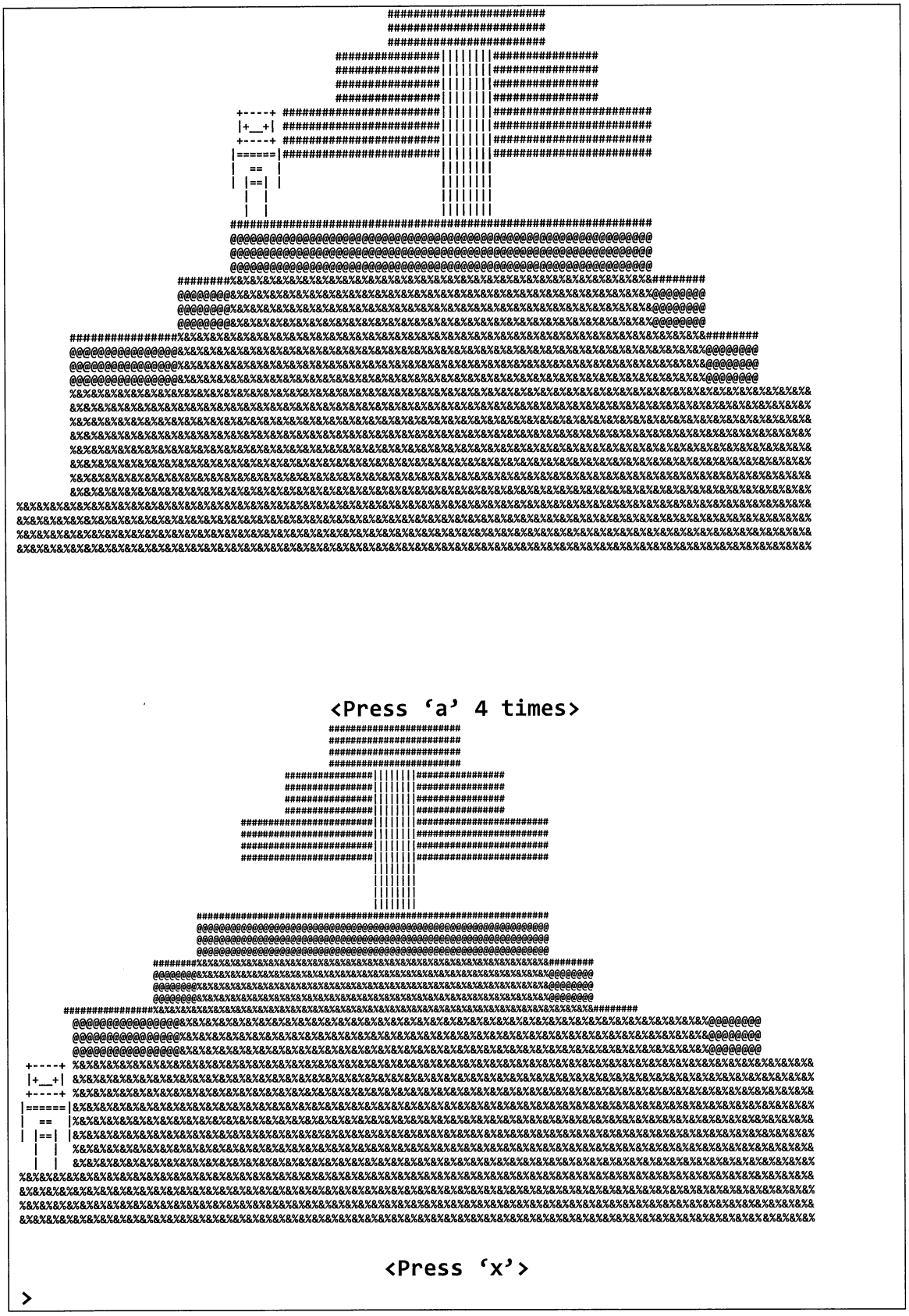
(a) **`onNormalKeyPress:`** The function will be called when a key is pressed. If `'a'` is pressed, the player moves left. Else if `'d'` is pressed the player moves right. Else if `'x'` is pressed exit the game.

(2) `Player.h`

(a) **`move:`** Move the player left or right. The player can go up 1 block. The player will fall if there is no block below. Need to check for x boundary.

The sample output of level 2:




<Press 'a' 4 times>
<Press 'x'>
>

E-3 MINE THE BLOCK (5 points)

Please complete all the required level 2 member functions in each above class declaration with the following requirements. For E-3, you need to implement the mining function. Use the arrow key to select the block. If no block is currently selected, select the block under the player. Otherwise, select the next block in the arrow key direction of the currently selected block. Also, if the player mines the block under itself, the player will fall.

(1) `Minecraft.h`

(a) **`onNormalKeyPress:`** The function will be called when a key is pressed. If `'f'` is pressed, mine the selected block. After the player moves, set **`selectedBlock = nullptr`**.

(b) **`onArrowKeyPress:`** The function will be called when the arrow key is pressed. If **`selectedBlock == nullptr`**, selected block under the player. Else, select the next block in the given direction of the currently selected block. The block also has to be inside the Player::mineDistance.

(2) `Player.h`

(a) **`mine:`** Mine a block. Call **`block->mine()`**. If the block below the player is mined, the player will fall. Return **`true`** if block is mined, else return **`false`**.

(3) `Block.h`

(a) **`mine:`** Mine the block, Increase the **`minedCount`** of the block. Return **`true`** if **`minedCount >= hardness`**, else return **`false`**.

(4) `Dirt.h`, `Leaf.h`, `Rock.h`, and **`Trunk.h`**

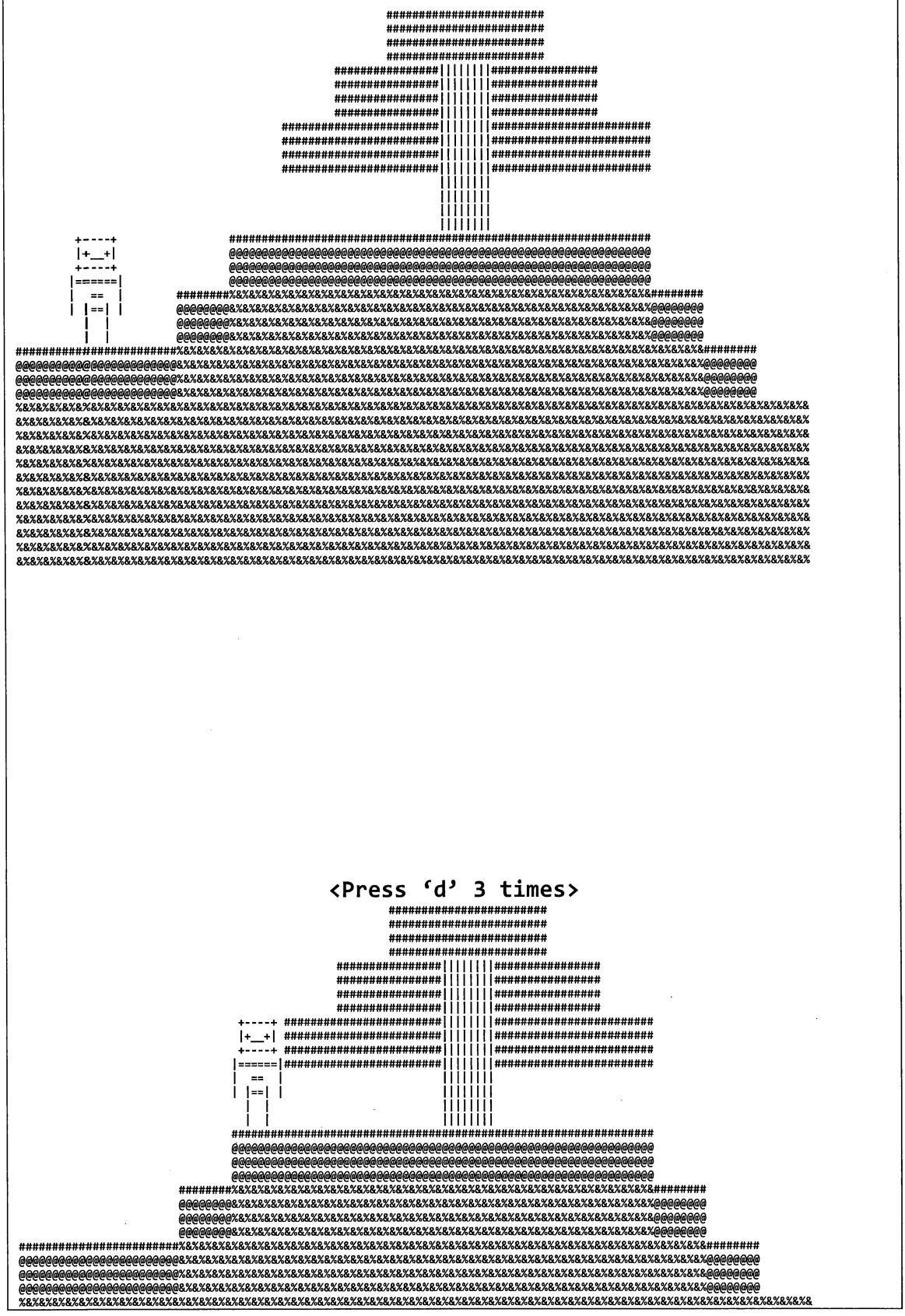
(a) `Render:` Render the block with **`selected == true`**.

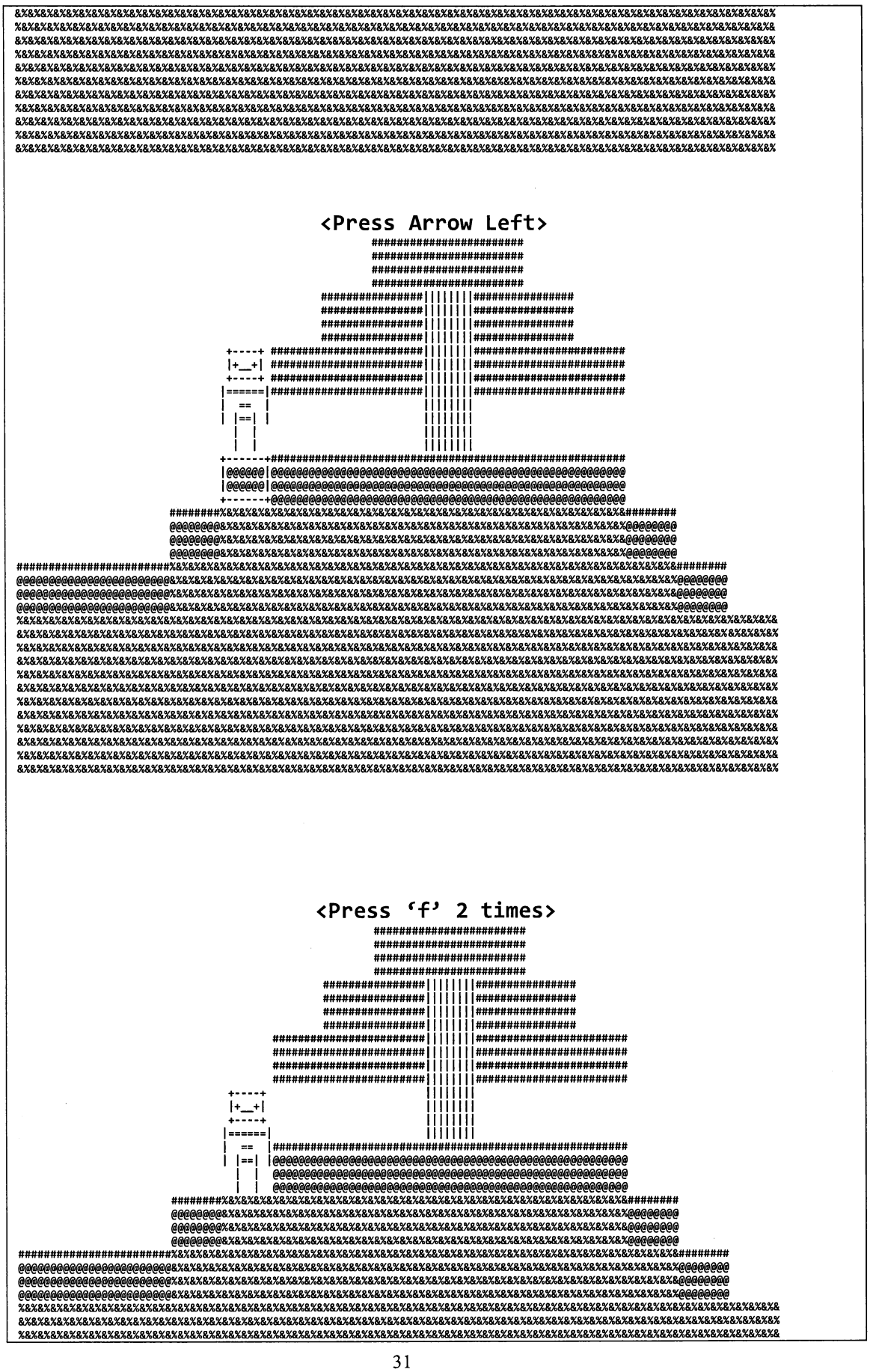
The selected images of each block

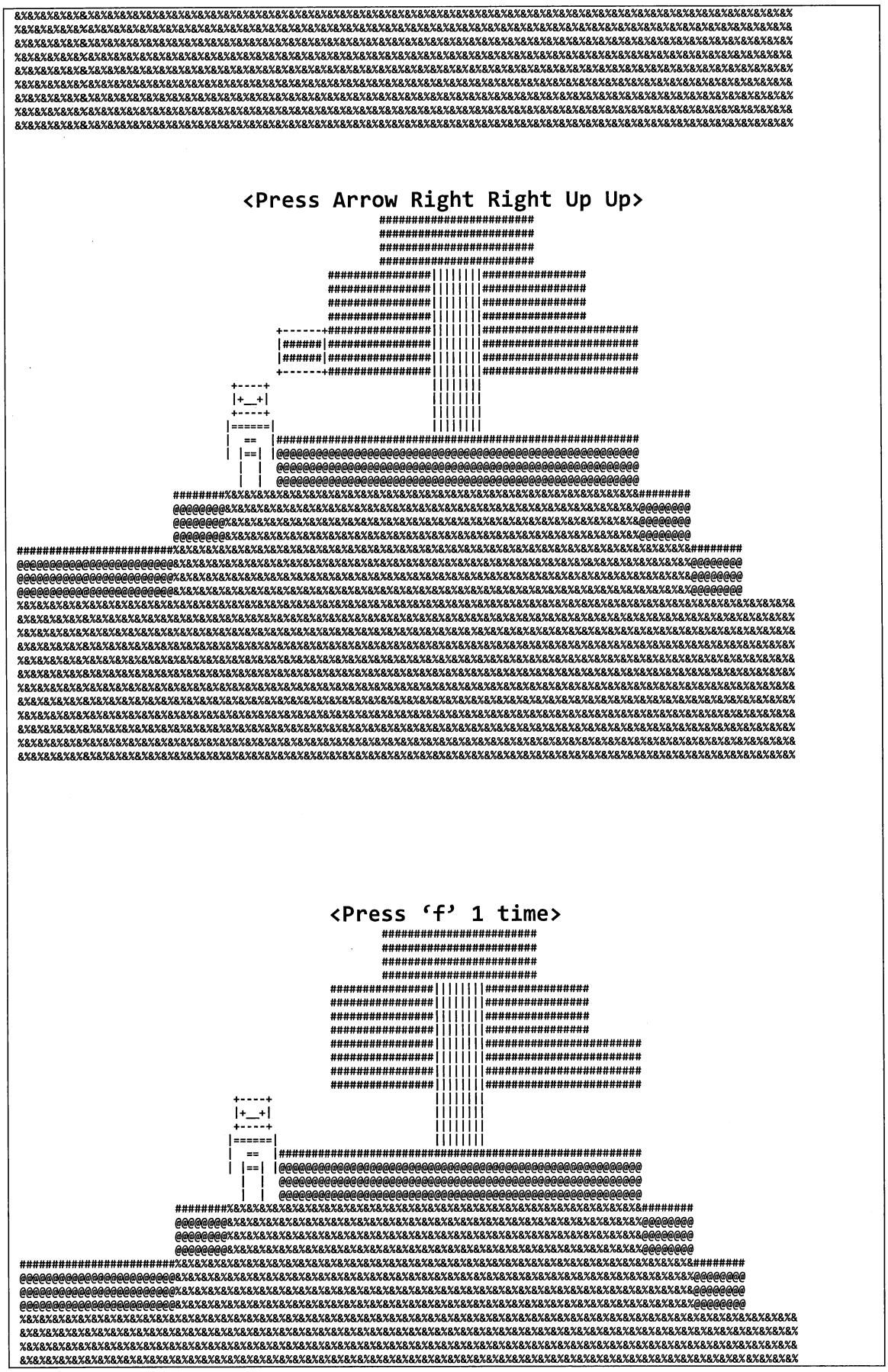
Dirt	Leaf	Trunk	Rock
hardness = 2	hardness = 1	hardness = 4	hardness = 6
+------+ |@@@@@@| |@@@@@@| +------+	+------+ |######| |######| +------+	+------+ |||||||| |||||||| +------+	+------+ |%&%&%&| |&%&%&%| +------+

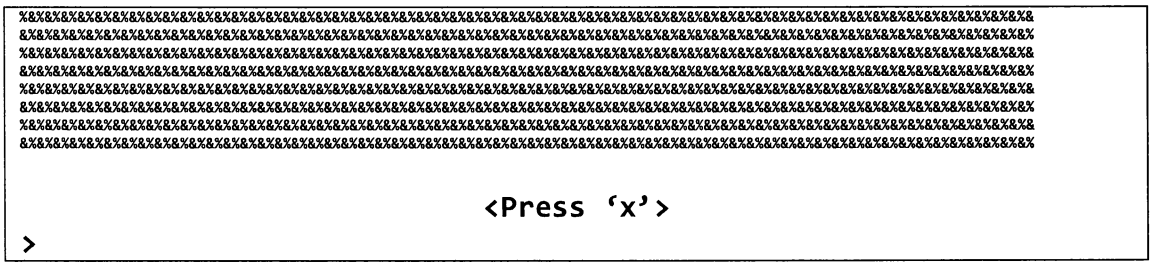
The sample output of level 3:

```
>./E-3
```




<Press Arrow Left>
<Press 'f' 2 times>


<Press Arrow Right Right Up Up>
<Press 'f' 1 time>



E-4 CUT DOWN A TREE (5 points)
Please cut down the tree in the middle.

【PROBLEM SET F】 (40%)

The online car rental system is a digital platform that enables customers to easily rent vehicles through the internet. It eliminates the need for physical visits to rental agencies and offers a streamlined process for reserving and managing rental cars. Customers can browse available vehicles, compare options, and make reservations online, while rental companies benefit from automated tasks and expanded market reach. The system provides convenience, efficiency, and a user-friendly experience for both customers and rental companies.

In this problem set, you must implement a simple Car Rental System with 3 vehicle types, customer, contract, and control system.

- ✔ The main programs are in `F-1.cpp`, `F-2.cpp`, `F-3.cpp`. **DO NOT MODIFY THE MAIN PROGRAMS.**
- ✔ You can modify and add proper prototypes to all the header files.
- ✔ To pass the test, your program cannot contain memory leaks. You can use the following command to test for memory leaks.

  ```
  > valgrind <your_execution_binary>
  ex: valgrind ./F-1
  ```

  Valgrind reference:
  https://web.stanford.edu/class/archive/cs/cs107/cs107.1234/resources/valgrind.html
- ✔ You can copy `F-1.cpp`, `F-2.cpp`, `F-3.cpp`, `Vehicle.h`, `Customer.h`, `RentalContract.h` `RentalSystem.h`, `Makefile`, `F-1`, `F-2`, and `F-3` from /home/share/final/F/.
- ✔ Demo command:

  Level 1 (20%):

  ```
  make F-1
  ./F-1
  ```

  Level 2 (10%):

  ```
  make F-2
  ./F-2
  ```

  Level 3 (10%):

  ```
  make F-3
  ./F-3
  ```

  Makefile usage:

  ```
  make: it will compile all the problems
  make clean: it will remove all the executable file
  ```

- ✔ You can execute `F-1`, `F-2`, and `F-3` to see what the result looks like.

F-1 COMPLETE THE FUNDAMENTAL SETUP (20 points)

Please complete all constructors, destructors, and the required level 1 member functions in each above class declaration with the following requirements. You also have to add class **Car**, class **Van**, and class **SUV** in **Car.h, Van.h,** and **SUV.h**, the above classes all inherited from class **Vehicle.**

(5) RentalSystem.h

- **(a) Constructor:** Create Rentalsystem.
- **(b) Destructor:** Delete all vehicles and customers information.
- **(c) add_customer:** Add new customer into the vector of customer.
- **(d) add_vehicle:** Add new vehicle into the vector of vehicle.
- **(e) add_account_value:** Deposit into the corresponding account.
- **(f) rent_car:** Rent a vehicle within the time interval [start_date, end_date], you can make a deposit at the same time. Please note that you will be charged once you make the reservation.
- **(g) return_car:** Return the vehicle. If you exceed the original rental period, you will be charged a penalty fee.
- **(h) generate_report:** Print out the information.

(6) RentalContract.h

- **(a) Constructor:** Initialize member variables, and call member function **start_of_contract**.
- **(b) start_of_contract:** create a contract between the customer and the vehicle, customer will be charged and the vehicle state will switch to unavailable.
- **(c) end_of_contract:** end of the contract between the customer and the vehicle, customer will be penalized if turn in the car late and the vehicle state will switch back to available.

(7) Customer.h

- **(a) Constructor:** Initialize member variables.
- **(b) rent_vehicle:**
  - i. It needs two arguments, the first one is the rented vehicle and the second one is the rental time duration.
  - ii. Compute the charge and update the account_balance.
- **(c) return_vehicle:**
  - i. It needs two arguments, the first one is the rented vehicle and the second one is the late day.
  - ii. Compute the penalty and update the account_balance.
- **(d) add_value:** Make a deposit.

(8) Vehicle.cpp

(a) Constructor: Initialize member variables.

(b) calculate_rental_price:

i. It needs two arguments, the first one is the rental time duration and the second one is the customer.

ii. charge = rental time duration * vehicle price

(c) calculate_penalty_fee:

i. It needs two arguments, the first one is the late day and the second one is the customer.

ii. penalty = late day * vehicle price * 0.1

(9) Car.h, Van.h, and **SUV.h**

(a) Constructor: Set the seat number for each type of vehicle.

(b) The properties of each vehicle are as followed

Car	SUV	Van
seats = 5	seats = 7	seats = 9

The sample output of level 1:

```
>make F-1
>./F-1
<<Customers>>
Name: Tom       Liscense: 511070        Balance: 10100
Name: Jack      Liscense: 511238        Balance: 1000
Name: Benson    Liscense: 511699        Balance: 0
<<Vehicles>>
Type: Toyota_ALTIS      Seats: 5        Price: 3200     Available: 1
Type: Ford_Tourneo      Seats: 9        Price: 5200     Available: 1
Type: Honda_FIT Seats: 5        Price: 2930     Available: 1
Type: Toyota_RAV4       Seats: 7        Price: 4200     Available: 1

Trade in 06/05
<Rent>
        Customer: Tom   Price: 3200     #Time: 2
        10100 - 6400 = 3700
<Rent>
        Customer: Mark  Price: 4200     #Time: 3
        20000 - 12600 = 7400
<<Customers>>
Name: Tom       Liscense: 511070        Balance: 3700
Name: Jack      Liscense: 511238        Balance: 1000
Name: Benson    Liscense: 511699        Balance: 0
Name: Mark      Liscense: 510123        Balance: 7400
<<Vehicles>>
Type: Toyota_ALTIS      Seats: 5        Price: 3200     Available: 0
```

```
Type: Ford_Tourneo      Seats: 9        Price: 5200     Available: 1
Type: Honda_FIT Seats: 5        Price: 2930     Available: 1
Type: Toyota_RAV4       Seats: 7        Price: 4200     Available: 0

Trade in 06/07
<Return>
        Customer: Mark  Price: 4200     #Late: 0
        7400 - 0 = 7400
<<Customers>>
Name: Tom       Liscense: 511070        Balance: 3700
Name: Jack      Liscense: 511238        Balance: 1000
Name: Benson    Liscense: 511699        Balance: 0
Name: Mark      Liscense: 510123        Balance: 7400
<<Vehicles>>
Type: Toyota_ALTIS      Seats: 5        Price: 3200     Available: 0
Type: Ford_Tourneo      Seats: 9        Price: 5200     Available: 1
Type: Honda_FIT Seats: 5        Price: 2930     Available: 1
Type: Toyota_RAV4       Seats: 7        Price: 4200     Available: 1

Trade in 06/08
<Return>
        Customer: Tom   Price: 3200     #Late: 2
        3700 - 640 = 3060
<<Customers>>
Name: Tom       Liscense: 511070        Balance: 3060
Name: Jack      Liscense: 511238        Balance: 1000
Name: Benson    Liscense: 511699        Balance: 0
Name: Mark      Liscense: 510123        Balance: 7400
<<Vehicles>>
Type: Toyota_ALTIS      Seats: 5        Price: 3200     Available: 1
Type: Ford_Tourneo      Seats: 9        Price: 5200     Available: 1
Type: Honda_FIT Seats: 5        Price: 2930     Available: 1
Type: Toyota_RAV4       Seats: 7        Price: 4200     Available: 1
>
```

F-2 COMPLETE THE DISCOUNT SYSTEM (10 points)

Please complete all the required level 2 member functions in each above class declaration with the following requirements. For F-2, you need to implement the discount system according to the membership level.

(3) **Vehicle.h**

(a) **calculate_rental_price:** Turn it into a virtual function that will be defined in each derived class.

(b) **calculate_penalty_fee:** Turn it into a virtual function that will be defined in each derived class.

(4) **Car.h, Van.h,** and **SUV.h**

(a) **calculate_rental_price:** Compute according to the rental time duration, vehicle price, and discount level.

(b) **calculate_penalty_fee:** Compute according to the late day, vehicle price, and penalty level.

(c) **The properties of each vehicle are as followed**

Equation				
Rental Price Computation = #rent_day * price * D Penalty Fee Computation = #late_day * price * P				
Car				
Bonus_point	<2	>=2	>=4	>=6
D	1	0.90	0.86	0.80
P	0.1	0.09	0.08	0.07
SUV				
Bonus_point	<3	>=3	>=5	>=10
D	1	0.95	0.90	0.85
P	0.15	0.12	0.10	0.08
Van				
Bonus_point	<2	>=2	>=5	>=8
D	1	0.80	0.75	0.70
P	0.20	0.15	0.12	0.10

(5) **Customer.h**

(a) **point_manipulate:** Manipulate Bonus_point.

i. +1: make a reservation (before charge computation)

ii. -1: return the vehicle late (after penalty computation)

(b) **rent_vehicle, return_vehicle:** Called **point_manipulate**.

The sample output of level 2:

```
>make F-2
>./F-2
<<Customers>>
Name: Tom       Liscense: 511070        Point: 3        Balance:
20100
Name: Jack      Liscense: 511238        Point: 8        Balance:
1000
Name: Benson    Liscense: 511699        Point: 10       Balance: 0
<<Vehicles>>
Type: Toyota_ALTIS      Seats: 5        Price: 3200     Available: 1
Type: Honda_FIT Seats: 5        Price: 2930     Available: 1
Type: Toyota_ALTIS      Seats: 5        Price: 3200     Available: 1
Type: Luxgen_M7 Seats: 7        Price: 4600     Available: 1
Type: Ford_KUGA Seats: 5        Price: 3000     Available: 1
Type: Ford_Tourneo      Seats: 9        Price: 5200     Available: 1
Type: Toyota_GRANVIA    Seats: 9        Price: 7600     Available: 1
Type: Toyota_SIENTA     Seats: 7        Price: 3900     Available: 1
Type: Toyota_RAV4       Seats: 7        Price: 4200     Available: 1

Trade in 06/28
<Rent>
        Customer: Tom   Price: 2930     #Time: 4        Point: 4
D:0.86
        20100 - 10079.2 = 10020.8
<Rent>
        Customer: Benson        Price: 7600     #Time: 2
Point: 11       D:0.7
        20000 - 10640 = 9360
<<Customers>>
Name: Tom       Liscense: 511070        Point: 4        Balance:
10020.8
Name: Jack      Liscense: 511238        Point: 8        Balance:
1000
Name: Benson    Liscense: 511699        Point: 11       Balance:
9360
<<Vehicles>>
Type: Toyota_ALTIS      Seats: 5        Price: 3200     Available: 1
Type: Honda_FIT Seats: 5        Price: 2930     Available: 0
Type: Toyota_ALTIS      Seats: 5        Price: 3200     Available: 1
Type: Luxgen_M7 Seats: 7        Price: 4600     Available: 1
Type: Ford_KUGA Seats: 5        Price: 3000     Available: 1
Type: Ford_Tourneo      Seats: 9        Price: 5200     Available: 1
Type: Toyota_GRANVIA    Seats: 9        Price: 7600     Available: 0
Type: Toyota_SIENTA     Seats: 7        Price: 3900     Available: 1
Type: Toyota_RAV4       Seats: 7        Price: 4200     Available: 1

Trade in 06/30
<Return>
```

```
        Customer: Benson        Price: 7600     #Late: 1
Point: 11       P:0.1
        9360 - 760 = 8600
<Rent>
        Customer: Jack  Price: 5200     #Time: 3        Point: 9
D:0.7
        31000 - 10920 = 20080
<<Customers>>
Name: Tom       Liscense: 511070        Point: 4        Balance:
10020.8
Name: Jack      Liscense: 511238        Point: 9        Balance:
20080
Name: Benson    Liscense: 511699        Point: 10       Balance:
8600
<<Vehicles>>
Type: Toyota_ALTIS      Seats: 5        Price: 3200     Available: 1
Type: Honda_FIT Seats: 5        Price: 2930     Available: 0
Type: Toyota_ALTIS      Seats: 5        Price: 3200     Available: 1
Type: Luxgen_M7 Seats: 7        Price: 4600     Available: 1
Type: Ford_KUGA Seats: 5        Price: 3000     Available: 1
Type: Ford_Tourneo      Seats: 9        Price: 5200     Available: 0
Type: Toyota_GRANVIA    Seats: 9        Price: 7600     Available: 1
Type: Toyota_SIENTA     Seats: 7        Price: 3900     Available: 1
Type: Toyota_RAV4       Seats: 7        Price: 4200     Available: 1

Trade in 07/01
<Return>
        Customer: Tom   Price: 2930     #Late: 0        Point: 4
P:0.08
        10020.8 - 0 = 10020.8
<Rent>
        Customer: Mark  Price: 7600     #Time: 2        Point: 1
D:1
        18000 - 15200 = 2800
<<Customers>>
Name: Tom       Liscense: 511070        Point: 4        Balance:
10020.8
Name: Jack      Liscense: 511238        Point: 9        Balance:
20080
Name: Benson    Liscense: 511699        Point: 10       Balance:
8600
Name: Mark      Liscense: 510123        Point: 1        Balance:
2800
<<Vehicles>>
Type: Toyota_ALTIS      Seats: 5        Price: 3200     Available: 1
Type: Honda_FIT Seats: 5        Price: 2930     Available: 1
Type: Toyota_ALTIS      Seats: 5        Price: 3200     Available: 1
Type: Luxgen_M7 Seats: 7        Price: 4600     Available: 1
Type: Ford_KUGA Seats: 5        Price: 3000     Available: 1
```

```
Type: Ford_Tourneo      Seats: 9        Price: 5200     Available: 0
Type: Toyota_GRANVIA    Seats: 9        Price: 7600     Available: 0
Type: Toyota_SIENTA     Seats: 7        Price: 3900     Available: 1
Type: Toyota_RAV4       Seats: 7        Price: 4200     Available: 1

Trade in 07/03
<Return>
        Customer: Jack  Price: 5200     #Late: 1        Point: 9
P:0.1
        20080 - 520 = 19560
<Return>
        Customer: Mark  Price: 7600     #Late: 1        Point: 1
P:0.2
        2800 - 1520 = 1280
<<Customers>>
Name: Tom       Liscense: 511070        Point: 4        Balance:
10020.8
Name: Jack      Liscense: 511238        Point: 8        Balance:
19560
Name: Benson    Liscense: 511699        Point: 10       Balance:
8600
Name: Mark      Liscense: 510123        Point: 0        Balance:
1280
<<Vehicles>>
Type: Toyota_ALTIS      Seats: 5        Price: 3200     Available: 1
Type: Honda_FIT Seats: 5        Price: 2930     Available: 1
Type: Toyota_ALTIS      Seats: 5        Price: 3200     Available: 1
Type: Luxgen_M7 Seats: 7        Price: 4600     Available: 1
Type: Ford_KUGA Seats: 5        Price: 3000     Available: 1
Type: Ford_Tourneo      Seats: 9        Price: 5200     Available: 1
Type: Toyota_GRANVIA    Seats: 9        Price: 7600     Available: 1
Type: Toyota_SIENTA     Seats: 7        Price: 3900     Available: 1
Type: Toyota_RAV4       Seats: 7        Price: 4200     Available: 1
>
```

F-3 CONSTRUCT THE RECOMMENDATION SYSTEM (10 points)

Please complete all the required level 3 member functions in each above class declaration with the following requirements. For F-3, you need to implement the recommendation system.

(5) RentalSystem.h

(a) car_recommendation:

i. It needs one arguments, the map includes all selection criterias. Also, only available vehicles will be recommanded.

ii. print out all suited candidates. (sort by price)

iii. return the cheapest one as best choice, else return NULL.

(b) generate_report (sorting)

i. Customers: sort by the order of Name

ii. Vehicles: sort by the order of Vehicle_Type -> Price -> Transmission_Type -> Brand -> Model.

(c) The key of criteria map and corresponding legal value.

Key	Value
Brand	Toyota, Honda, etc.
Vehicle	Car, SUV, Van
LPrice_Bound (lowest price)	A number i.e.3000
HPrice_Bound (highest price)	A number i.e.5000
Trans_Type	Auto, Manual

The sample output of level 3:

```
>make F-3
>./F-3
<<Customers>>
Name: Benson   Liscense: 511699      Point: 10     Balance: 0
Name: Jack     Liscense: 511238      Point: 8      Balance: 1000
Name: Tom      Liscense: 511070      Point: 3      Balance: 20100
<<Vehicles>>
Type: Honda_FIT Seats: 5      Price: 2930   Trans: Manual  Available: 1
Type: Ford_KUGA Seats: 5      Price: 3000   Trans: Manual  Available: 1
Type: Toyota_ALTIS    Seats: 5      Price: 3400   Trans: Auto    Available: 1
Type: Toyota_ALTIS2   Seats: 5      Price: 3400   Trans: Auto    Available: 1
Type: Toyota_SIENTA   Seats: 7      Price: 3300   Trans: Manual  Available: 1
Type: Toyota_RAV4     Seats: 7      Price: 4200   Trans: Auto    Available: 1
Type: Luxgen_M7 Seats: 7      Price: 4600   Trans: Auto    Available: 1
Type: Ford_Tourneo    Seats: 9      Price: 5200   Trans: Auto    Available: 1
Type: Toyota_GRANVIA  Seats: 9      Price: 7600   Trans: Auto    Available: 1

Trade in 06/28
<Rent>
    Customer: Tom  Price: 2930   #Time: 4      Point: 4      D:0.86
    20100 - 10079.2 = 10020.8
```

```
///Recommendation////
Type: Toyota_SIENTA    Seats: 7      Price: 3300    Trans: Manual   Available: 1
Type: Toyota_ALTIS     Seats: 5      Price: 3400    Trans: Auto     Available: 1
Type: Toyota_ALTIS2    Seats: 5      Price: 3400    Trans: Auto     Available: 1
Type: Toyota_RAV4      Seats: 7      Price: 4200    Trans: Auto     Available: 1
///Recommendation////
Best Choice:  Toyota_SIENTA

<Rent>
     Customer: Benson      Price: 3300    #Time: 2      Point: 11      D:0.85
     20000 - 5610 = 14390
<<Customers>>
Name: Benson    Liscense: 511699      Point: 11      Balance: 14390
Name: Jack      Liscense: 511238      Point: 8       Balance: 1000
Name: Tom       Liscense: 511070      Point: 4       Balance: 10020.8
<<Vehicles>>
Type: Honda_FIT Seats: 5      Price: 2930    Trans: Manual   Available: 0
Type: Ford_KUGA Seats: 5      Price: 3000    Trans: Manual   Available: 1
Type: Toyota_ALTIS     Seats: 5      Price: 3400    Trans: Auto     Available: 1
Type: Toyota_ALTIS2    Seats: 5      Price: 3400    Trans: Auto     Available: 1
Type: Toyota_SIENTA    Seats: 7      Price: 3300    Trans: Manual   Available: 0
Type: Toyota_RAV4      Seats: 7      Price: 4200    Trans: Auto     Available: 1
Type: Luxgen_M7 Seats: 7      Price: 4600    Trans: Auto     Available: 1
Type: Ford_Tourneo     Seats: 9      Price: 5200    Trans: Auto     Available: 1
Type: Toyota_GRANVIA   Seats: 9      Price: 7600    Trans: Auto     Available: 1

Trade in 06/30
<Return>
     Customer: Benson      Price: 3300    #Late: 1      Point: 11      P:0.08
     14390 - 264 = 14126
///Recommendation////
///Recommendation////
Doesn't find any suitable choice. Loosen your criteria and try again!

///Recommendation////
Type: Ford_KUGA Seats: 5      Price: 3000    Trans: Manual   Available: 1
Type: Toyota_SIENTA    Seats: 7      Price: 3300    Trans: Manual   Available: 1
Type: Toyota_ALTIS     Seats: 5      Price: 3400    Trans: Auto     Available: 1
Type: Toyota_ALTIS2    Seats: 5      Price: 3400    Trans: Auto     Available: 1
Type: Toyota_RAV4      Seats: 7      Price: 4200    Trans: Auto     Available: 1
Type: Luxgen_M7 Seats: 7      Price: 4600    Trans: Auto     Available: 1
Type: Ford_Tourneo     Seats: 9      Price: 5200    Trans: Auto     Available: 1
///Recommendation////
Best Choice:  Ford_KUGA

<Rent>
     Customer: Jack  Price: 3000    #Time: 3      Point: 9      D:0.8
```

```
    31000 - 7200 = 23800
<<Customers>>
Name: Benson   Liscense: 511699      Point: 10     Balance: 14126
Name: Jack     Liscense: 511238      Point: 9      Balance: 23800
Name: Tom      Liscense: 511070      Point: 4      Balance: 10020.8
<<Vehicles>>
Type: Honda_FIT Seats: 5      Price: 2930    Trans: Manual   Available: 0
Type: Ford_KUGA Seats: 5      Price: 3000    Trans: Manual   Available: 0
Type: Toyota_ALTIS    Seats: 5      Price: 3400    Trans: Auto    Available: 1
Type: Toyota_ALTIS2   Seats: 5      Price: 3400    Trans: Auto    Available: 1
Type: Toyota_SIENTA   Seats: 7      Price: 3300    Trans: Manual   Available: 1
Type: Toyota_RAV4     Seats: 7      Price: 4200    Trans: Auto    Available: 1
Type: Luxgen_M7 Seats: 7      Price: 4600    Trans: Auto    Available: 1
Type: Ford_Tourneo    Seats: 9      Price: 5200    Trans: Auto    Available: 1
Type: Toyota_GRANVIA  Seats: 9      Price: 7600    Trans: Auto    Available: 1

Trade in 07/01
<Return>
    Customer: Tom  Price: 2930    #Late: 0      Point: 4      P:0.08
    10020.8 - 0 = 10020.8
///Recommendation////
Type: Ford_Tourneo    Seats: 9      Price: 5200    Trans: Auto    Available: 1
///Recommendation////
Best Choice:  Ford_Tourneo

<Rent>
    Customer: Mark  Price: 5200    #Time: 2      Point: 1      D:1
    18000 - 10400 = 7600
<<Customers>>
Name: Benson   Liscense: 511699      Point: 10     Balance: 14126
Name: Jack     Liscense: 511238      Point: 9      Balance: 23800
Name: Mark     Liscense: 510123      Point: 1      Balance: 7600
Name: Tom      Liscense: 511070      Point: 4      Balance: 10020.8
<<Vehicles>>
Type: Honda_FIT Seats: 5      Price: 2930    Trans: Manual   Available: 1
Type: Ford_KUGA Seats: 5      Price: 3000    Trans: Manual   Available: 0
Type: Toyota_ALTIS    Seats: 5      Price: 3400    Trans: Auto    Available: 1
Type: Toyota_ALTIS2   Seats: 5      Price: 3400    Trans: Auto    Available: 1
Type: Toyota_SIENTA   Seats: 7      Price: 3300    Trans: Manual   Available: 1
Type: Toyota_RAV4     Seats: 7      Price: 4200    Trans: Auto    Available: 1
Type: Luxgen_M7 Seats: 7      Price: 4600    Trans: Auto    Available: 1
Type: Ford_Tourneo    Seats: 9      Price: 5200    Trans: Auto    Available: 0
Type: Toyota_GRANVIA  Seats: 9      Price: 7600    Trans: Auto    Available: 1

Trade in 07/03
<Return>
    Customer: Jack  Price: 3000    #Late: 1      Point: 9      P:0.07
```

```
    23800 - 210 = 23590
<Return>
    Customer: Mark  Price: 5200    #Late: 1      Point: 1      P:0.2
    7600 - 1040 = 6560
<<Customers>>
Name: Benson    Liscense: 511699      Point: 10     Balance: 14126
Name: Jack     Liscense: 511238      Point: 8      Balance: 23590
Name: Mark     Liscense: 510123      Point: 0      Balance: 6560
Name: Tom      Liscense: 511070      Point: 4      Balance: 10020.8
<<Vehicles>>
Type: Honda_FIT Seats: 5      Price: 2930    Trans: Manual   Available: 1
Type: Ford_KUGA Seats: 5      Price: 3000    Trans: Manual   Available: 1
Type: Toyota_ALTIS     Seats: 5      Price: 3400    Trans: Auto     Available: 1
Type: Toyota_ALTIS2    Seats: 5      Price: 3400    Trans: Auto     Available: 1
Type: Toyota_SIENTA    Seats: 7      Price: 3300    Trans: Manual   Available: 1
Type: Toyota_RAV4      Seats: 7      Price: 4200    Trans: Auto     Available: 1
Type: Luxgen_M7 Seats: 7      Price: 4600    Trans: Auto     Available: 1
Type: Ford_Tourneo     Seats: 9      Price: 5200    Trans: Auto     Available: 1
Type: Toyota_GRANVIA   Seats: 9      Price: 7600    Trans: Auto     Available: 1
>
```