1. (15%) Linux Command

Write a **single** command that can

(1) (2%) Copy a file named `main.cpp` to the directory `project1`. Both file and directory are assumed to exist and to be in the current working directory.

```
cp main.cpp project1
```

(2) (2%) Change the **current working directory** to its parent directory. (For example, if your current working directory is `/home/john/`, then its parent directory is `/home/`.)

```
cd ..
```

(3) (2%) Read a long text file named `data.txt` in an interactive read-only mode.

```
cat data.txt -l
```

(4) (3%) List the content **including the hidden content** of the home directory with a general command that works for any user. Suppose you are not in your home directory.

```
ls -a ~
```

(5) (3%) Delete a directory along with all of the files and subdirectories inside the directory. Suppose that the name of the directory is `dir`.

```
rm -r dir
```

(6) (3%) Compare two files `code.cpp` and `code.old` to see whether they are different. Instead of showing the result on the screen, save it in a file called `result.txt`.

```
diff code.cpp code.old >> result.txt
```

2. (12%) Building a C++ program

On ECE2800J Online Judge server, a submission of Project 7 includes 4 files. `p7.cpp`, `p7.h`, `circuit.cpp`, and `circuit.h`. They are built into an executable (p7) using the following (incomplete) `Makefile`. Line numbers are labeled on the left. For blanks (3a), (4a) and (4b), see below.

```
1|all: p7
2|
3|p7: p7.o circuit.o
4|    g++ -o p7.o circuit.o p7
5|
6|p7.o: p7.cpp
7|    g++ _____(3a)_____
8|
9|circuit.o: circuit.cpp
10|   g++ _____(3b)_____
11|
12|clean:
13|   rm _____(4a)_____
14|
```

(1) (2%) There is a syntax error in this file. Where is it? How will you fix it?

It's in line 4, it should be g++ p7.cpp -o p7 p7.o circuit.o

(2) (1%) How could we build `p7` with this `Makefile`?

using commands: ~~make p7.o make circuit.o~~ make p7

(3) (2%) Fill in the blanks (3a) and (3b) so that we are able to build `p7` using the command you provide in (2).

(3a): -c p7.cpp

(3b): -c circuit.cpp

(4) (2%) The target `clean` removes the previously built executable and object files. Fill in the blank (4a) using ~~no more~~ than 10 characters, spaces included.

(4a): *.o *.exe -o

(5) (1%) How could we remove the previously built executable and object files with this `Makefile`?

make clean

(6) (4%) Suppose `circuit.h` contains the following code:

```
// beginning of circuit.h
class Circuit_t {
// class content omitted
};
// end of circuit.h
```

What is a potential issue of the above code? How to fix it?

~~Reincluded~~ The .h file ~~xxxx~~ may be multi-included.

We fix it by adding header guard:

```
~~#ifndif~~ #ifndef CIRCUIT-H
            #def CIRCUIT-H
              // content of circuit.h
            #endif
```

-0.5

3. (22%) C++ Programming Language

(1) (3%) Given the following C++ statements, what are the final values of x, z, and r?

```
int x = 8;
int &r = x;
r = 5;
int z = 3;
r = z;
r = 2;
```

r为x引用. x变5. z变3 x变3 x变2.

2 ; 3 ; 2

(2) (2%) Given the following C++ statements, what is the final value of z?

```
typedef enum {APPLE, ORANGE, PEAR, BANANA, STRAWBERRY} Fruit_t;
const int NUM = 12;
int value[NUM];
for(int i = 0; i < NUM; i++)
    value[i] = 3*i+1;
Fruit_t fruit = BANANA;
int z = value[fruit];
```

0 1 2 3 12 12

value: 1 4 7 10 13 ...

10.

(3) (2%) Define a function pointer that could point to the following function:

```
double volume(double length, double width, double height);
```

double (*funcptr)(double length, double width, double height);

(4) (4%) Suppose that variable v is an int and variable p is a pointer to int. For each of the following expressions, please tell if it is an lvalue or an rvalue.

(a) 2*v : rvalue

(b) *p : lvalue

(c) &v : rvalue

(d) &p : ~~lvalue~~ rvalue

(5) (4%) Consider the following C++ statements. What is the output?

```
void func(int x) {
    if(x < 4) func(++x);
    cout << x << ",";
}
```

1 → 4 3 2 1

```
int main() {
    func(1);
    return 0;
}
```

4, 3, 2, 1, -3

(6) (7%) What is the output of the following code?

```
void f(double x) {
    cout << "f begins" << endl;
    throw x;
    cout << "f ends" << endl;
}

void g(double x) {
    cout << "g begins" << endl;
    try {
        f(x);
    }
    catch(double e) {
        cout << "Error in g double: val = " << e << endl;
    }
    cout << "g ends" << endl;
}

int main() {
    try {
        cout << "Try block entered" << endl;
        g(10);
        cout << "Leaving try block" << endl;
    }
    catch (int e) {
        cout << "Error int: val = " << e << endl;
    }
    catch (double e) {
        cout << "Error in main doule: val = " << e << endl;
    }
    cout << "After catch block" << endl;
    return 0;
}
```

4. (8%) Recursion and Trees

Recall that in Project 2, we have defined a binary tree type. A **binary tree** is well-formed if:

- It is an empty tree, or
- It consists of an integer element, called the **root element**, plus two children, called the **left subtree** and the **right subtree**, each of which is a well-formed binary tree.

Based on this definition, we further define a **product tree**. A *binary tree* is called a product tree if and only if:

1. It is an empty tree, or
2. It is a leaf (i.e., a tree with a single element), or
3. Its left and right subtrees are both *non-empty* product trees and its root element equals the product of the root elements of both subtrees.

To help you write the required function, suppose that we have defined a type `tree_t` for representing a binary tree. Also, the following methods for `tree_t` are provided:

```
bool tree_isEmpty(tree_t tree);
// EFFECTS: returns true if "tree" is empty, false otherwise

int tree_elt(tree_t tree);
// REQUIRES: "tree" is not empty
// EFFECTS: returns the root element of "tree"

tree_t tree_left(tree_t tree);
// REQUIRES: "tree" is not empty
// EFFECTS: returns the left subtree of "tree"

tree_t tree_right(tree_t tree);
// REQUIRES: "tree" is not empty
// EFFECTS: returns the right subtree of "tree"
```

Given the definition of a product tree, write the following function

```
bool is_product_tree(tree_t tree);
// EFFECTS: returns true if "tree" is a product tree,
//          false otherwise .
//
// For example, for the following tree
//       24
//      /  \
//     8    3
//    / \
//   2   4
//  / \
// 0
// 24 is the product of 8 and 3, 8 is the product of 2 and 4, and
// some other requirements of a product tree are satisfied. Thus,
// it is a product tree and the function should return true.
```

int product;

tree_t inOrderTraversal(tree_t root){

if (! tree_isEmpty) inOrderTraversal(tree_left)

实现: product *= tree_elt(root)

inOrderTraversal(product , root)

fix:

Notes:

- You are allowed to write helper functions. If they are needed, please also write the implementations of these helper functions, **including specification comments**. For simplicity, you can omit the specification comments for the function `is_product_tree`.
- Your solution must be **recursive**. You may not use loops, global / static variables, and goto-s. You may use branches.

You can only write your solution on the following lines.

```
1| bool is_product_tree (tree_t tree) {
2|     if (tree_is Empty(tree)) return true;
3|   return is_product_tree (tree_left(tree)) &&
4|             is_product_tree (tree_right(tree));
5| if (tree_elt (tree_left(tree)) * tree_elt (tree_right(tree))
6|     == tree_elt (tree)) return true;
7| else return false; }
8|
9|
10|
11|
12|
13|
14|
15|
16|
17|
18|
19|
20|
21|
22|
```

5. (43%) Splitting a Number

Given three integers n, b, and s, where $n \geq 1, b \geq 0, s \geq 0$, you are asked to print **all** different vectors $(x_1, x_2, \ldots, x_n)$ of length n such that for all $1 \leq i \leq n, x_i \in \{0,1,\ldots,b\}$, and $\sum_{i=1}^{n} x_i = s$. For example, given $n = 3, b = 2, s = 4$, you should print the following vectors, which satisfy the above conditions:

(0, 2, 2), (1, 1, 2), (1, 2, 1), (2, 0, 2), (2, 1, 1), (2, 2, 0).

Note that we do not enforce a particular order on these vectors. As long as your program prints **all** of them, it is good.

Note that if there is no such a vector, your program prints **nothing**.

You are going to represent a vector using the `list_t` type from Project 2. Recall that a **list** is well-formed if:

- It is an empty list, or
- It is an integer followed by a well-formed list.

The type `list_t` is used to represent a list. Also, the following methods for `list_t` are provided:

```
bool list_isEmpty(list_t list);
// EFFECTS: returns true if "list" is empty, false otherwise

list_t list_make();
// EFFECTS: returns an empty list

list_t list_prepend(int elt, list_t list);
// EFFECTS: given "list", make a new list consisting of
//          the new element "elt" followed by the elements
//          of the original "list"

List_t list_append(list_t list, int elt);
// EFFECTS: given "list", make a new list consisting of
//          the elements of the original "list" followed by
//          the new element "elt"

int list_first(list_t list);
// REQUIRES: "list" is not empty
// EFFECTS: returns the first element of "list"

list_t list_rest(list_t list);
// REQUIRES: "list" is not empty
// EFFECTS: returns the list containing all but the first
//          element of "list"

void list_print(list_t list);
// MODIFIES: cout
// EFFECTS: prints "list" followed by a new line
```

Besides, you are also given the following function

```
int atoi(char* str);
// REQUIRES: "str" is a null-terminated C string
// EFFECTS: if "str" is a string representing an integer,
//          returns that integer
```

Suppose the function that achieves the above requirement is

```
void print_split(int n, int b, int s)
// REQUIRES: n >= 1, b >= 0, s >= 0
// EFFECTS: print all different vectors of length n such that each
//          entry is an integer in the range [0, b] and the sum of
//          all the entries in the vector is s.
```

(1) (5%) What are the output vectors by calling function `print_split(3, 3, 6)`?

(3,2,1), (3,1,2), (2,3,1), (2,1,3), (1,2,3), (1,3,2)

(2) (8%) The program you are asked to write takes the values for n, b, and s from command line. To be more concrete, suppose that your program is named as `split`. If you run it as `./split 3 2 4`, then $n = 3, b = 2, s = 4$. You are asked to write the `main` function. The main function needs to check the following errors:

- Missing arguments.
- An input value outside its required range.

If any of the above error happens, the `main` function issues a corresponding error message and returns. Please feel free to decide the error message. If none of the above error happens, you can assume that the inputs are valid. Then, the `main` function calls the function `print_split` to print all the vectors.

(3) (12%) Testing your function is important. Write 4 boundary test cases for the function `print_split`. Each case should test a **different** boundary condition. For each test case, you must provide a description of the test case, the expected behavior for a correct implementation of the function, and why the case is a boundary test case.

① ② ③

① : print_split(n, b, 0), in this case, the output is confirmed, and it should be (0,0···0) (n elements). Reason: n is of their smallest value.

② print_split(1, b, s), in this case, the output is (s) if b ≥ s, no output if b < s. Reason: n is of its smallest value.

③ print_split(n, 0, s), in this case, the output is (0,0···,0) (n elements) if s=0, no output if s≠0. Reason: b is of its smallest value.

(4) (18%) Implement the function `print_split`. You are asked to use the `list_t` type to represent a vector and use the operations provided above. You can write helper functions. If so, also give their implementations together with the specification comments. For simplicity, you can omit the specification comments for the function `print_split`. **Hint**: you may want to define a recursive helper function.