EE205 Project 2 Precept

Oct 3rd, 2017

Jiin Woo
jidoll333@kaist.ac.kr

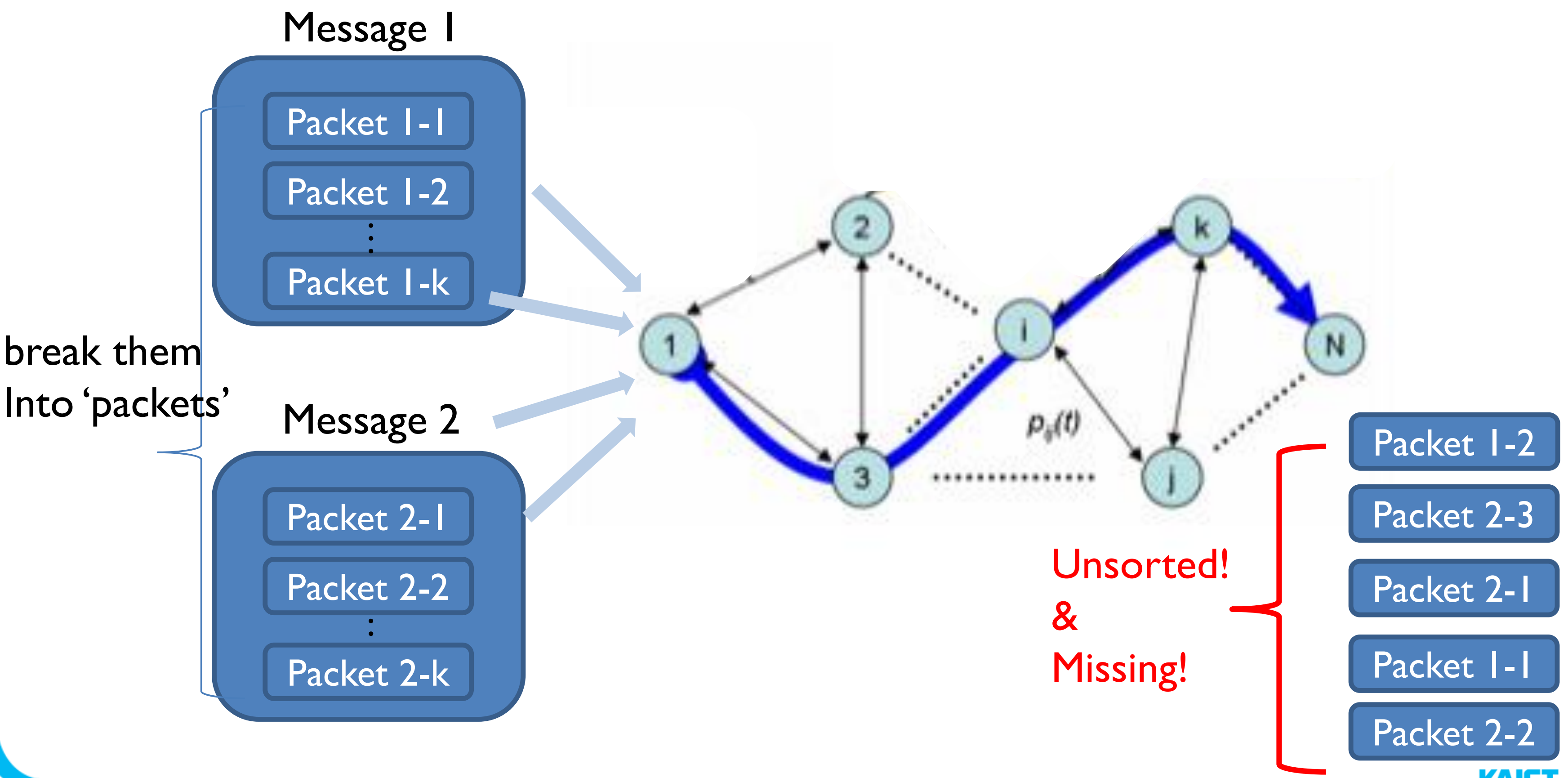
Project 2 Introduction

- Application of linked list structure
  - Singly linked list
  - Doubly linked list

- Apply to where?
  - Packet network system

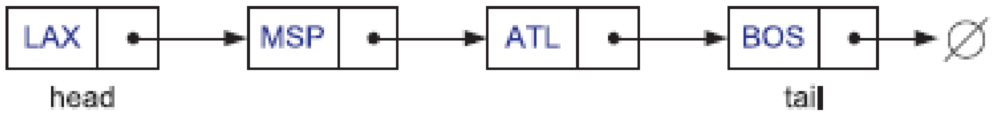
Message Receiver in Packet Network

- Modern computer networks take messages and break them into smaller units called 'packets'.
- Some packets is lost and reordered at destination.

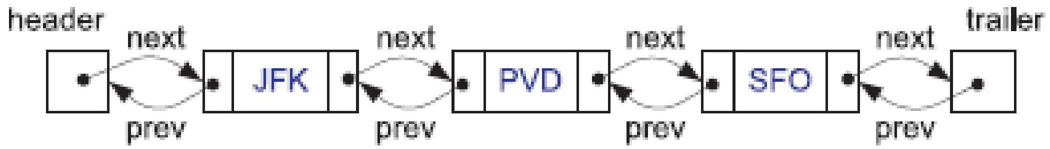


Message Receiver in Packet Network

- What the receiver should do?
  - Reconstruct original messages from partially received packets
  - Sort the packets & find missing packets
- How?
  - Use linked list data structures
    - Singly linked list



    - Doubly linked list



Project Objective

- Project #a
  - Implement linked list structures (Singly & Doubly)

- Project #b
  - Implement receivers using the implemented linked list in project #a
  - Compare the running time of receivers using singly and doubly linked list for each

Note: project #a is the stepping stone of project #b

File structure

- Baseline codes
  - single_list.h
  - single_list.cpp
  - double_list.h
  - double_list.cpp
  - single_receiver.cpp
  - double_receiver.cpp

Project #a

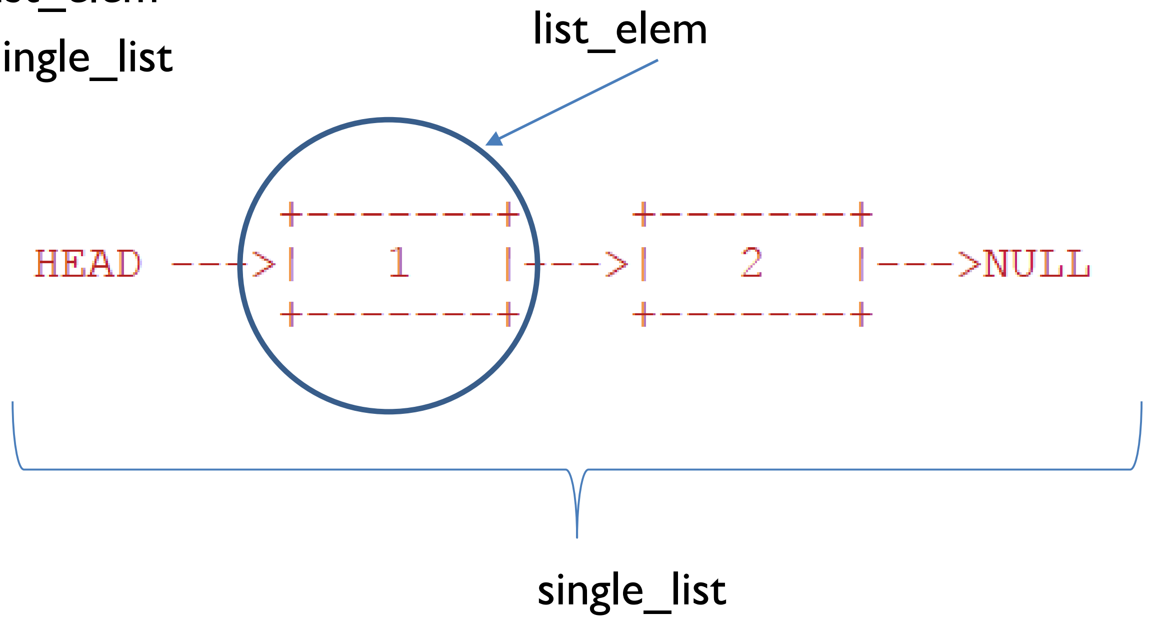
Project #b

Header files(.h) : define the basic structure and functions of linked lists.
Cpp files(.cpp) : skeleton codes for the functions you should implement.

Note: Modify only cpp files

Project #a (1)

- Singly linked list implementation
  - Basic structures
    - list_elem
    - single_list

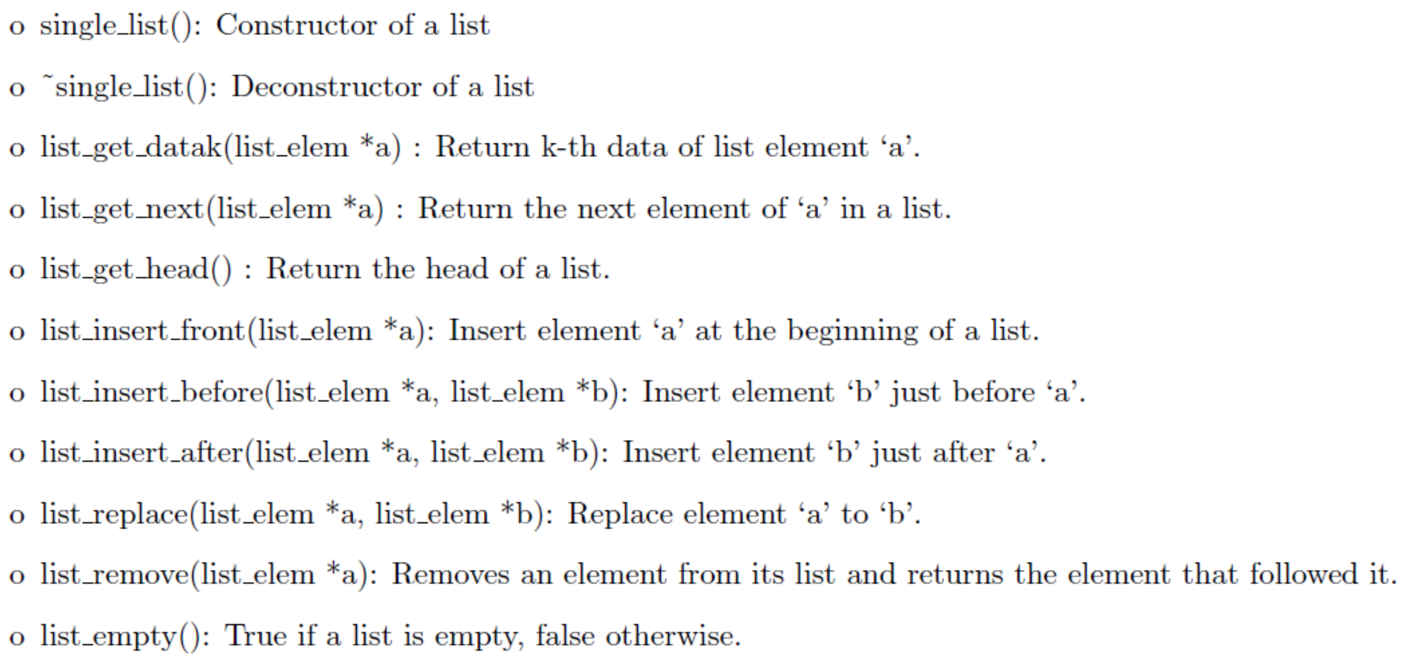


More detailed definition of the structure is in single_list.h

Project #a (2)

- Singly linked list implementation
  - To do
    - Complete functions in single_list.cpp (NOT single_list.h)

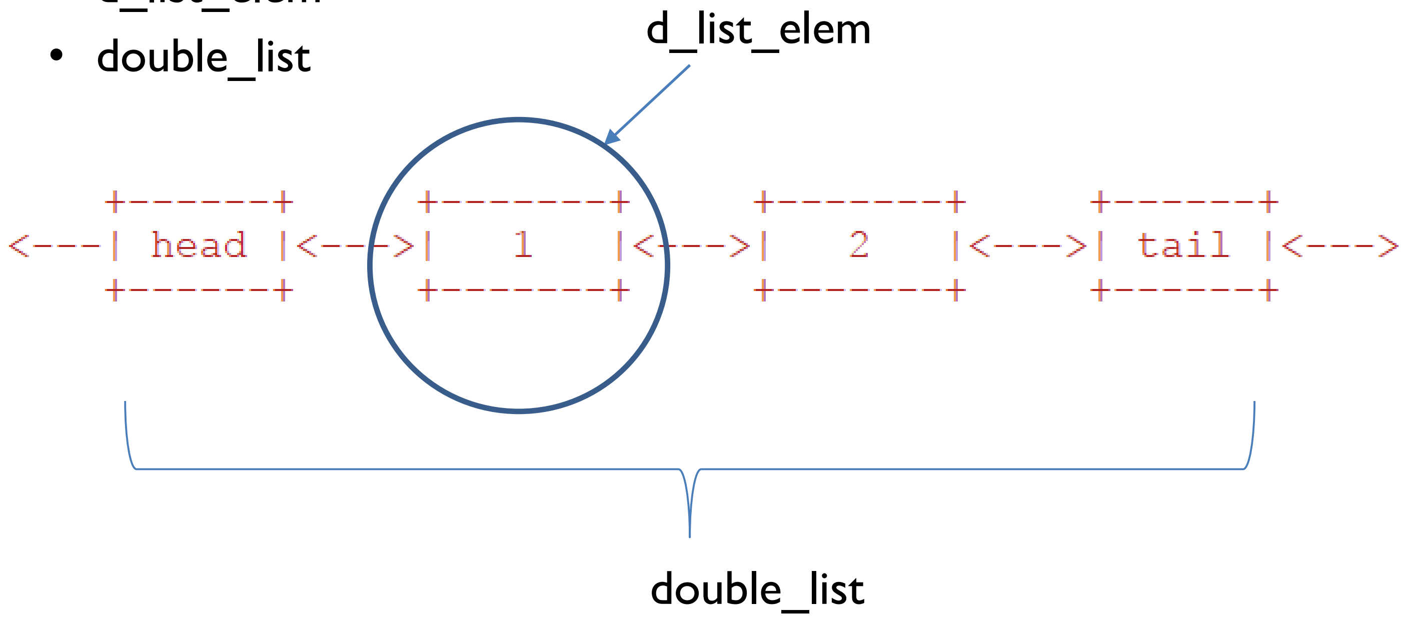
Basic operations necessary for singly linked lists.



- o single_list(): Constructor of a list
- o ˜single_list(): Deconstructor of a list
- o list_get_datak(list_elem *a) : Return k-th data of list element 'a'.
- o list_get_next(list_elem *a) : Return the next element of 'a' in a list.
- o list_get_head() : Return the head of a list.
- o list_insert_front(list_elem *a): Insert element 'a' at the beginning of a list.
- o list_insert_before(list_elem *a, list_elem *b): Insert element 'b' just before 'a'.
- o list_insert_after(list_elem *a, list_elem *b): Insert element 'b' just after 'a'.
- o list_replace(list_elem *a, list_elem *b): Replace element 'a' to 'b'.
- o list_remove(list_elem *a): Removes an element from its list and returns the element that followed it.
- o list_empty(): True if a list is empty, false otherwise.

Project #a (3)

- Doubly linked list implementation
  - Basic structures
    - d_list_elem
    - double_list

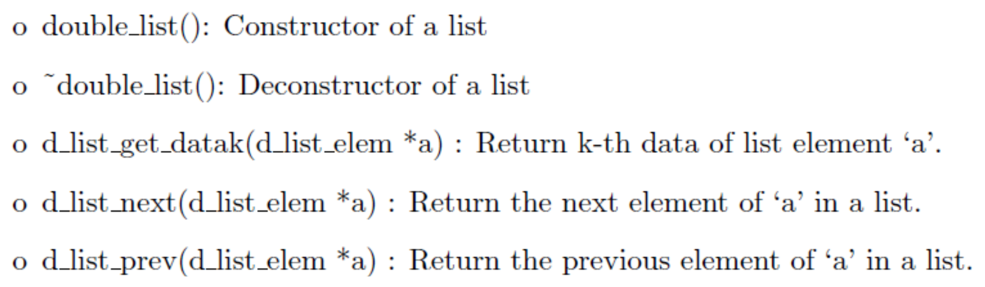


```
     +-------+    +--------+    +--------+    +-------+
<---| head  |<--->|   1    |<--->|   2    |<--->| tail  |<--->
     +-------+    +--------+    +--------+    +-------+
```

More detailed definition of the structure is in double_list.h

Project #a (4)

- Doubly linked list implementation
  - To do
    - Complete functions in double_list.cpp (NOT double_list.h)



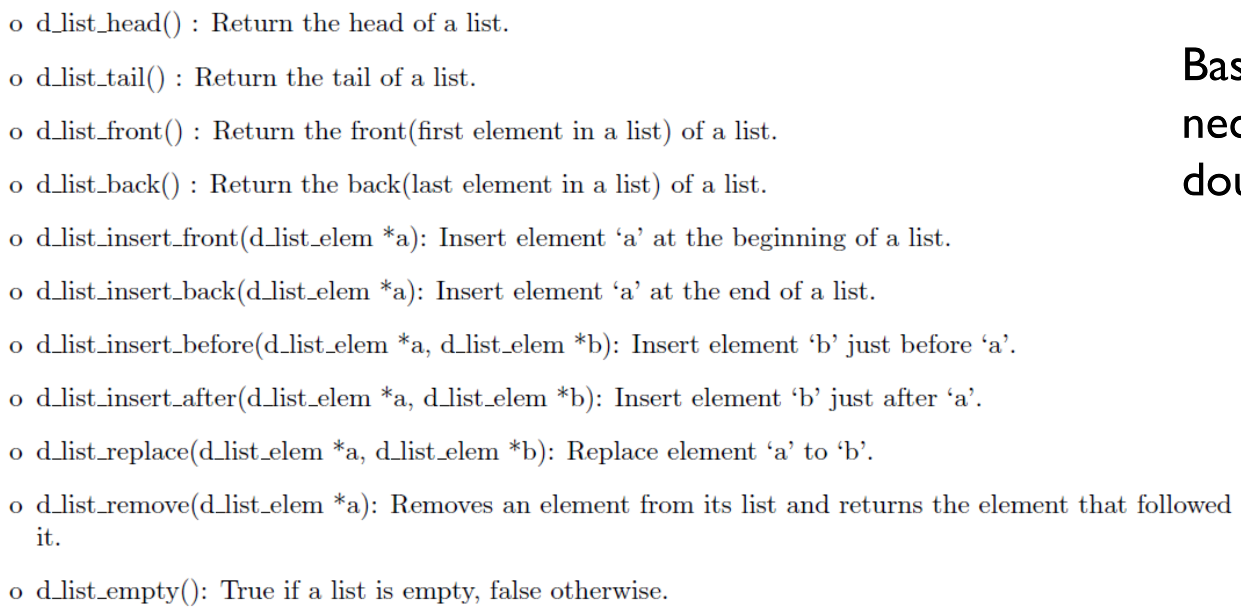
o double_list(): Constructor of a list

o ˜double_list(): Deconstructor of a list

o d_list_get_datak(d_list_elem *a) : Return k-th data of list element 'a'.

o d_list_next(d_list_elem *a) : Return the next element of 'a' in a list.

o d_list_prev(d_list_elem *a) : Return the previous element of 'a' in a list.



o d_list_head() : Return the head of a list.

o d_list_tail() : Return the tail of a list.

o d_list_front() : Return the front(first element in a list) of a list.

o d_list_back() : Return the back(last element in a list) of a list.

o d_list_insert_front(d_list_elem *a): Insert element 'a' at the beginning of a list.

o d_list_insert_back(d_list_elem *a): Insert element 'a' at the end of a list.

o d_list_insert_before(d_list_elem *a, d_list_elem *b): Insert element 'b' just before 'a'.

o d_list_insert_after(d_list_elem *a, d_list_elem *b): Insert element 'b' just after 'a'.

o d_list_replace(d_list_elem *a, d_list_elem *b): Replace element 'a' to 'b'.

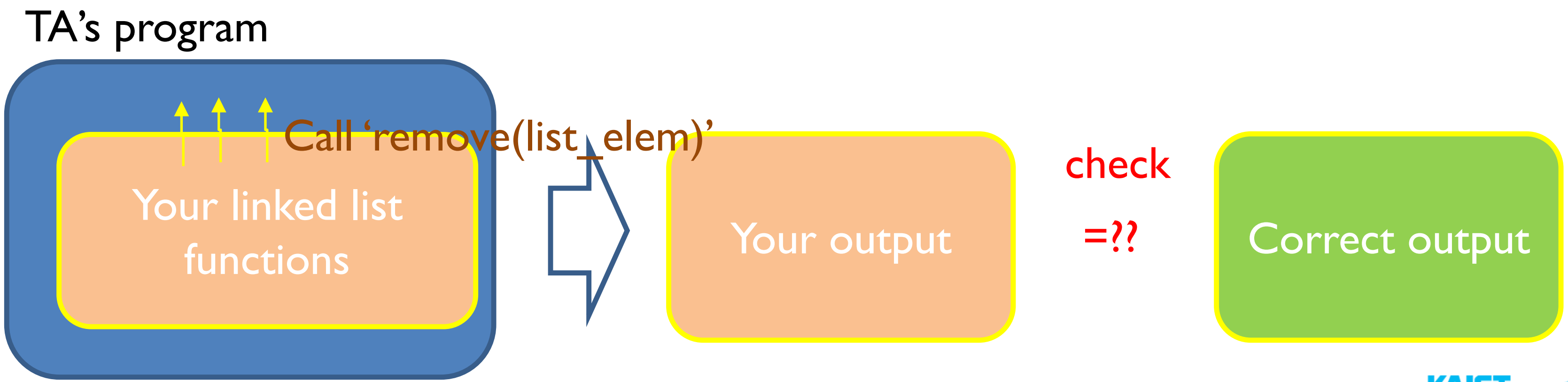
o d_list_remove(d_list_elem *a): Removes an element from its list and returns the element that followed it.

o d_list_empty(): True if a list is empty, false otherwise.

Basic operations necessary for doubly linked lists.

Project #a (5)

- Submission
  - [EE205 project2a]20XXXXXX.tar.gz
    - Single_list.cpp, double_list.cpp, readme
- Evaluation
  - TA's program will call your linked list functions.
  - TA's program will do some tasks which require correctly operating linked list functions.
  - We will check the output of TA's program is correct.



Project #b (1)

- Implement two receiver programs(singly/doubly) satisfying
  1. Get unsorted packets from an input file
     ```
     ./single_receiver input_file output_file
     ./double_receiver input_file output_file
     ```
  2. Sort packets in order of msg number and packet number
  3. Print sorted packets and missed packets to an output file
  4. Print running time.

Use linked list structures and functions implemented in pj #a.

Note:
Single_receiver: receiver using singly linked list
Double_receiver: receiver using doubly linked list

More detailed description in the project documentation.

Project #b (2)

- Input -> Output format

Unsorted packets in input file

```
message_number    packet_number    word
```

```
message_number    packet_number    word
```

```
message_number    packet_number    word
```

1. Sorted packets

```
- Message i
word i,1
word i,2
...
word i,n
- End Message i
```

If some packet is missing?
Print this instead of the packet!

```
WARNING: packet j of message i is missing!
```

2. Running time

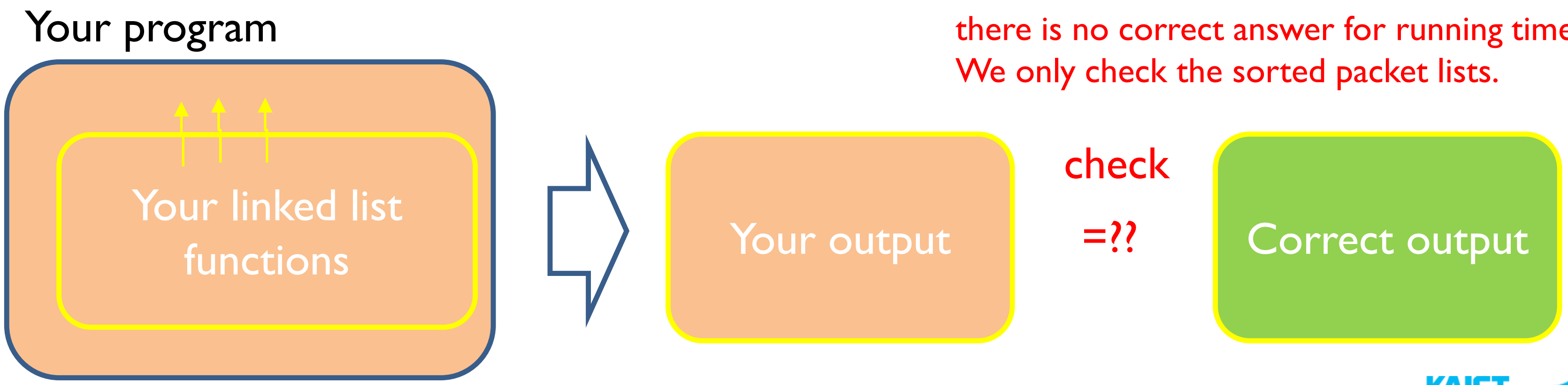
```
Running Time: [t] s.
```

More detailed description in the project documentation.

Project #b (3)

- Submission
  - [EE205 project2b]20XXXXXX.tar.gz
    - single_list.cpp, double_list.cpp, single_receiver, double_receiver, readme
- Evaluation
  - Several input files which test various functionality of your receiver program
  - The output of your program will be compared to the correct output
  - Running time should be written at the end of output file
  - The report about comparison between running time of the single receiver and double receiver

Note:
there is no correct answer for running time
We only check the sorted packet lists.



Tips

- Start early
- Read the project documentation carefully
- Check out the baseline code and get started
- Refer to the text book and google
- Ask ambiguous things about the project in KLMS

lanada

Thank you

Good luck and Have a nice 추석 :)