Final review

CSE 30: Computer Organization and Systems Programming

Diba Mirza
Dept. of Computer Science and Engineering
University of California, San Diego
Announcements

- Final review, Sun Dec 6\textsuperscript{th} from 11am to 12:20pm
  WLH 2001
Doubly linked lists

- Consider a node in a doubly linked list
  
  ```c
  typedef struct ListNode ListNode;
  struct ListNode {
    int data;
    ListNode *next;
    ListNode *prev;
  };
  ```

In C

```c
    ListNode *head;
    ```
Doubly linked lists

▶ Write recursive code to reverse the order of elements in the list
ListNode * reverse(ListNode *aNode) ;
Doubly linked lists

ListNode * reverse(ListNode *aNode) {
    if (aNode == NULL) return aNode;

    ListNode * tmp = aNode->next; // swap next & prev of aNode
    aNode->next = aNode->prev;
    aNode->prev = tmp;

    reverse (aNode->prev); // recurse

    return aNode;
}

This is incorrect - why?
Reverse Recursively first try

ListNode * reverse(ListNode *aNode) {
    if (aNode == NULL) return aNode;
    ListNode * tmp = aNode->next;
    aNode->next = aNode->prev;
    aNode->prev = tmp;
    reverse(aNode->prev);
}

Which of the following is true about the above implementation:
A. Doesn't swap the next and prev pointers of the last node in the list
B. Its not recursive, although it produces the right result.
C. It doesn't return the correct value after reversing the list
D. It correctly reverses the list and returns the new head
Reverse Recursively 2\textsuperscript{nd} try

```c
ListNode * reverse(ListNode *aNode) {
    if (aNode == NULL) return aNode;
    ListNode * tmp = aNode->next;
    aNode->next = aNode->prev;
    aNode->prev = tmp;
    if (aNode->prev == NULL) // Reached the last node
        return aNode; // return aNode; /* Reversed the last node */
    reverse(aNode->prev);
}
```

Which of the following is true about the above implementation:

A. Doesn’t swap the next and prev pointers of the last node in the list
B. It’s not recursive, although it produces the right result.
C. It doesn’t return the correct value after reversing the list
D. It correctly reverses the list and returns the new head
ListNode * reverse(ListNode *aNode) {
    if (aNode == NULL) return aNode;
    ListNode * tmp = aNode->next;
    aNode->next = aNode->prev;
    aNode->prev = tmp;
    if (aNode->prev == NULL)
        return aNode;
    return reverse(aNode->prev);
}

\[
\begin{array}{c|c|c|c}
\hline
\text{data} & 10 & \text{data} & 20 \\
\text{next} & \text{} & \text{next} & \text{} \\
\text{prev} & \text{} & \text{prev} & \text{} \\
\hline
\end{array}
\]
Complex declarations in C

```c
int ***p;
int (**pa)[];
int (**pf)[];
int *(*pap)[];
int (*paa)[][];
int *(*pf)();
int **app[];
int *(*apa[])[];
int *aap [][];
int aaa[][[]];
```
Summary

- How is data stored in memory
- How are programs translated to machine language
- How do high-level programs run on ARM processors
- Got a bug?
  - just dive in deep until you find it.

And remember…….

GDB is your friend