
# COEN 11
# Advanced Programming

## Lab 2


# Lab 2 – Task List

- You will create a program to manage a list of tasks.
- Code is due on Friday of week 2, at midnight


# Lab 2

- The task list is created interactively with the following commands
  - 1 <task> <hour>
    - Adds the *task and hour* specified to the bottom of the list
    - A task may have more than one word, so enter the info one per line
  - 2
    - Shows the list, task and hour, from oldest to newest
  - 3 <hour>
    - Shows the tasks at the given *hour*, from oldest to newest
    - The *hour* may be in the same or in the next line
  - 0
    - Quits


# Requirements

- Global variables:
  - 2 arrays (tasks and hours)
    - The arrays should be able to hold 10 tasks
  - counter
    - The counter shows the number of tasks in the arrays

- Main function: loop forever accepting commands
  - Use the code provided in lab 1

- Add 3 new functions:
  - insert
  - list
  - list_hour


# List Mechanism

- Your list should stay in the oldest-to-newest order
  - Always insert a new entry at the bottom, which is indicated by the counter
  - No loops are necessary


# How to read a line

- Reading a line from the keyboard
  - To read a task, use scanf with fpurge

```
fpurge (stdin);
scanf ("%[^'\n']", line); // line is a char array
```

→ fpurge empties the buffer which may contain a '\n'

→ scanf will read everything but the next '\n'


# Before You Submit and Demo

- Make sure your code work!
  - Test all the possible behaviors
    - Insert different combinations and check both listing functions
      - Insert different tasks with different times
      - Insert same task with same time
      - Insert different tasks with same time
      - Insert same tasks with different times
    - Make sure your code works in extreme conditions
      - Check the listing functions when the list is empty and full
      - Try to insert 11 elements


# You Are In Charge of Your Code!

- If the code is not compiling, follow the compiler's output
  - Get rid of warnings too
- If your code is not working, try to find the error on your own
  - Debugging is like solving a mystery
  - Put several **printf** with numbers along the code to understand where the code is going

```
printf ("1\n");
...
printf ("2\n");
...
```

  - Output values of variables to check if they have what they should have


# Grading

- Pre-lab
  - Flowchart of the insert function – 10%
- When you are done
  - Add lab2 to the makefile
  - Demo to the TA – 30%
  - Submit your code to Camino – 60%