Lecture

- There will no lectures in week 07.
- Prepare for Tuesday, February 12, 2013:
  2.9, 2.10, 2.14, 2.19, and the following C exercises

C programming exercises - Part 1

From the online C tutorial `http://www.cprogramming.com/tutorial.html#ctutorial` read the parts Structures, Arrays, C-style Strings, File I/O, and Typecasting. In the tutorial session focus on pointers:

What will be the output of this C language program?

```
void inc_ptr(int* p){
  (*p)++;
  return;
}
int main(){
  int *p;
  *p=12;
  inc_ptr(p);
  printf("%d", *p);
  return 0;
}
```

What will be the output of this C language program?

```
void inc_ptr(int* p){
  (*p)++;
  return;
}

int main(){
  int o=12;
  int *p;
  p=&o;
  inc_ptr(p);
  printf("%d", *p);
  return 0;
}
```

What will be the output of this C language program?

```
#include <stdio.h>

main()
{
  int i, array[10];
  int *ip, *a1;
  int **ipp;

  ip = &i;
  ipp = &ip;
  a1 = &(array[1]);

  for (i = 0; i < 10; i++) array[i] = i;

  i = 11;

  printf("ip: 0x%x, &ip: 0x%x, array: 0x%x\n", ip, &ip, array);
  printf("\n");


  printf("&i: 0x%x\n", &i);
  printf("ipp: 0x%x, *ipp: 0x%x, **ipp: 0x%x\n", ipp, *ipp, **ipp);
  printf("\n");
  printf("a1: 0x%x, *a1: 0x%x\n", a1, *a1);

  a1 += 4;
  *a1 = 500;

  for (i = 0; i < 10; i++) {
    printf("%d ", array[i]);
  }
  printf("\n");
}
```

C programming exercises - Part 2

From the online C tutorial `http://www.cprogramming.com/tutorial.html#ctutorial` read the parts Command line arguments, Linked Lists, Recursion, Variable argument lists, Binary Trees. In the tutorial session focus on the following exercises:

- Write a non-recursive version of a function, which reverses a string `s` in place.
- Write a recursive version of the same function, which reverses a string `s` in place.
- Write a C function `int linearSearch(int a[], int first, int last, int key)` that performs a linear search in an array a with key `key` with `first` and `last` being the smallest and largest index of an element in the array.
- Write a C function that performs a binary search in an array a with key `key`.
- Make a recursive implementation of the binary search.