COMP10002 Workshop Week 4

Outlook:

1	Functions & scopes: Ex 6.2
2	Pointers & Functions
3	Arrays, pointers and arrays. Discuss Ex 7.3.
4	Implement 6.9 and 7.4
github this week	• `insertion_sort.c` and data • skeleton for all exercises this week

Scopes, local & global variables

```
#include <stdio.h>
int ga, gb;
int f( int, int );

int main (int argc, char *argv[]){
    int ma, mb;
    ... ga=1; ....
    ... f(ma, mb)...
    ... ga++...
    return 0;
}

int f(int f1, int f2) {
    int fa, fb;
    ...ga+gb+f1...
    return ...;
}
```

scope of **argc**, **argv**, **ma**, and **mb.**

scope of **f1**, **f2**, **fa**, and **fb**

scope of **f**

scope of **ga** and **gb**

6.2

***6.2**: For each of the 3 marked points, write down a list of all of the program-declared variables and functions that are in scope at that point, and for each identifier, its type. Don't forget* **main**, **argc**, **argv**. *Where there are more than one choice of a given name, be sure to indicate which one you are referring to.*

```
int bill(int jack, int jane);
double jane(double dick, int fred, double dave);

int trev;

int main(int argc, char *argv[]) {
     double beth;
     int pete, bill;      /* -- point #1 -- */
     return 0;
}

int bill (int jack, int jane) {
     int mary;
     double zack;         /* -- point #2 -- */
     return 0;
}

double jane(double dick, int fred, double dave) {
     double trev;         /* -- point #3 -- */
     return 0.0;
}
```

Function: can it have more than one output? how?

functions can have more than one outputs

1	`int n;`	
2		
3	`scanf("%d", &n);`	What were sent to `scanf` ?
4		How many output scanf give in this case?
5		
6	`print("n= %d\n", n);`	Why not **`&n`** here?

What is `&n` ? what's its data type?

Data type: “pointer to int”, “address of int”

`int a= 18;`	18 **a**	**a** is a location in the memory, interpreted as an **int**, with value of 18
`int *pa;`	? **pa** 18 **a**	**pa** is an **int pointer**, it can hold the address of an **int**
`pa= &a;`	&a → 18 **pa** **a**	**pa** now holds the address of **a**, or, it "points" to **a**. **pa** is another method to access **a** using **pa** alone, we can both ***get*** and ***change*** the value of **a**

`*pa= *pa + 1;`	&a → 19
	pa **a**
		These statements are all equivalent:
		`*pa = *pa + 1;`
		`a = a + 1;`
		
		`*pa = a + 1;`
		`a = *pa + 1;`

pointers as function parameters

Pointers can be used to change the value of variables *indirectly*

Example: Function call in line 4 leads to the change of value of sum and product.

```
1   int main(...) {
2     int a=2, b=4, sum, product;
3        ...
4     sAndP(a, b, &sum, &product);
5
6     printf("sum=%d",                sum);
7     printf("prod=%d",                   product);
8        ...
9   }
11
12  void sAndP(int m, int n, int *ps, int *pp ) {
13    // now ps has the value &sum, *ps is sum
      *ps = m + n ;
      *pp = m * n ;
14  }
```

&sum is copied to ps, and so *ps is the same as sum

Quiz 1

With the fragment:

```
int x= 10;
f(&x);
```

which function below will set `x` *to zero?*

A:

```
int f(int n) {
   return 0;
}
```

B:

```
void f( int *n) {
  &n= 0;
}
```

C:

```
void f (int *n) {
   n= 0;
}
```

D:

```
void f( int *n) {
  *n= 0;
}
```

Quiz 2

Given function:

```
void f(int a, int *b) {
    a= 1;
    *b = 2;
}
```

what are values of `m` *and* `n` *after the following fragment:*

```
m= 5;
n= 10;
f(m, &n);
```

A) 5 and 10	B) 1 and 2	C) 5 and 2	D) 1 and 10

Arrays

(one-dimensional) arrays

1	The statement `int A[5];`	
2	is equivalent to declaring 5 variables, each is of data type **int**:	A[0] A[1] A[2] A[3] A[4]
3	`A[0] = 10;`	10
4	`i= 2;` `A[i]= 20;`	10 20

5	`for (i=0; i<5; i++) {`
	`  A[i]= i*i;`
	`}`
		0 1 4 9 16

6	`for (i=0; i<5; i++) {`
	`  scanf ( "%d", &A[i] );`
	`}`	

Arrays

With declaration:

```
#define MAX_N 1000
int A[MAX_N]= {10, 20, 30}, n= 3, i= 1;
int *p= A;
```

- `A[i]` is just an `int` variable
- `A` is: a constant? an array? a pointer? an address?
- `p` is a variable, and now `p` can be used as a substitute for `A`

- Passing array to a function? We can just pass:
  - `A`: the array name, and
  - `n`: the actual number of elements in the array `A`.

Functions with arrays: examples

Write a function that:

- computes sum of a `float` array
- double the value of all elements of an `int` array

How to use these functions?

Arrays: example using insertion_sort.c from github

Insertion sort algorithm:

Input: an array of `n` (say, `int`) elements

Output: the same array, but with elements re-arranged in some (say, increasing) order.

At home: copy `insertion_sort.c` from `github` and play with it.

7.3: Modify so that only distinct values are retained after sorting
Use **c7.3.c** from **github** for skeleton and hints

```
for (i= 1; i<n ; i++) {
   /* swap A[i] left into correct position */
   for (j= i-1; j>=0 && A[j+1]<A[j]; j--) {
      /* not there yet */
      int_swap( &A[j] , &A[j+1] );
   }
}
```

```
H: ./7.3.exe
Enter up to 1000 values, ^Z to end
1 8 15 3 17 12 4 8 4
^Z
9 value read into array
Before:  1  8 15  3 17 12  4  8  4
After :  1  3  4  8 12 15 17
```

Lab: Ex 6.9

Target: Read integer amounts of cents between 0 and 999. Print out the coin changes, using coins from $2 to 5c. For example, for cents = 393, the printout should be:

200c, 100c, 50c, 20c, 20c, 5c

(for simplicity, we print “200c” instead of “$2”, but you can try $2!)

Note: valid coins are 200c, 100c, 50c, 20c, 10c, and 5c

Design: supposing we want to write a function for print changes

Target: … For example, for cents = 393, the printout should be:
200c, 100c, 50c, 20c, 20c, 5c

Lab: Ex 6.9 – use 6.9.c from github

Target: … For example, for cents = 393, the printout should be:
200c, 100c, 50c, 20c, 20c, 5c
(for simplicity, we print "200c" instead of "$2", but you can try "$2")

Write & use functions:

`int round_to_5(int cents)`
that returns the value rounded off to the nearest multiple of 5.

`int try_one_coin(int *pcents, int coin)`
that returns number of coins with face value "`coin`",
does the corresponding printout,
and also reduces the value of "`*pcents`" accordingly.

`void print_change(int cents)`
that employs `try_one_coin` to print out the coin
changes for the amount "`cents`".

Lab

Finish implementing Ex 6.9.

Implement Ex 7.4(output values and frequencies of an array). Note that we need 2 solutions for this task: one trivial and general, and the other one which imposes some constraints on input data.

Use `github` as you wish:

- `e7.4.Sol1.c` : skeleton for solution 1
- `e7.4.Sol2.c` : skeleton for solution 2
- Think: The 2nd solution better? When? Why?