DALHOUSIE UNIVERSITY *Inspiring Minds*

CSCI 2132
Midterm 2 Solutions

Term: Fall 2018 (Sep4-Dec4)

1. (10 points) True-false questions: 2 points each. Justification is not necessary, but brief justification may be helpful if correct.

a) (2 points) In the C programming language, the following two pairs of `scanf` format strings are equivalent: `"%c:%c"` and `"%c: %c"`

Solution: False. If the input is '`4: 5`' (with space after colon), the result would be different.

b) (2 points) The `-g` option of `gcc` is used to produce object code only from a C source file.

Solution: False. The option `-g` is used to add symbolic information to the code. The option `-c` will produce object code only from a source file.

c) (2 points) The following C code is valid: `int i; double d=5.1; i = d;`

Solution: True. The code is valid and the value of `i` will be `5` due to implicit type conversion.

d) (2 points) The function parameters and function local variables are stored on the call stack.

Solution: True.

e) (2 points) After executing '`int a[10]={1};`' the value of `a[10]` is not defined.

Solution: True. The value of `a[10]` is not defined and should not be used because it is out of range.

2. (12 points) Multiple-choice. Circle the correct answer to the question.

a) (3 points) Which of the following statements is FALSE about processes?

A. We can start a process in background by using character '`&`' in the command line.

B. A foreground process can print to the terminal.

C. A background process can read input from the keyboard.

D. A foreground process can run in the same time (concurrently) as a background process.

> **Solution:** $\boxed{\textbf{C.}}$ is False. A background process cannot read from the keyboard. The other statements are true.

---

b) (3 points) Which phase is NOT part of the Waterfall Model of software development life cycle?

A. Verification

B. Requirement Analysis

C. Prototype Development

D. Design

> **Solution:** $\boxed{\textbf{C.}}$ The Prototype Development is a part of the different model: the Rapid Prototyping, not the Waterfall Model.

---

c) (3 points) After the following code:
`int a[10]={10,20,30}; int *p; p=&a[2]; p -= 1; --(*--p);`
the array `a` will start with the following values or an error is generated:

A. `{9,19,29}`

B. `{10,20,27}`

C. Invalid pointer operation (possibly Segmentation-fault error)

D. `{9,20,30}`

> **Solution:** $\boxed{\textbf{D.}}$ After `p=&a[2];` `p` is pointing to `a[2]`, after `p -= 1;` `p` is pointing to `a[1]`, and after `--(*--p);` `p` will point to `a[0]` and decrease it by 1.

---

d) (3 points) The fork system call is used in the following situation:

A. Creation of a new process.

B. Creation of a new sub-directory.

C. Execution of a conditional statement.

D. Creation of a new stack frame.

> **Solution:** $\boxed{\textbf{A.}}$ The other statements are not true.

3. (12 points) Give concise answers.

a) (4 points) Briefly describe `gdb` commands `break`, `step`, and `next`.

Solution: The `gdb` command `break` is used to set up a breakpoint; i.e., a place where program will pause execution and give us a chance to examine the variable values and other elements of program state.

The `gdb` command `step` is used to step execution of a program line by line. When execution comes across a function call, the step will step into the function call.

The `next` command is similar to `step`, however when a function call is executed, the `next` command will step over it; i.e., it will execute it as one step instead of going into the function.

b) (4 points) If we have the declarations '`int *p, a[10] = {1};`' briefly explain the meaning of the statement: '`p = a+2;`'. Is there another way to write this statement?

Solution: (2 points) The meaning of the statement '`p = a+2;`' is the same as '`p = &a[2];`', which means that we set the pointer `p` to point to the third element of the array `a`.

(2 points) Another way to write the statement is: '`p = &a[2];`'

c) (4 points) If we execute MergeSort on array `{4,1,3,7,6,2,5,8}`, how many times will the function 'merge' be executed? What will be sub-arrays that are merged during the last execution of the 'merge' function? (List the values of sub-arrays.)

Solution: (1 points) The function would be executed 7 times in total.

(3 points) During the last execution of merge, the following sub-arrays would be merged: (1,3,4,7) and (2,5,6,8).

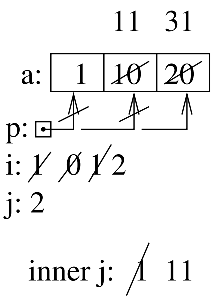
4. (8 points) Code snippets.

a) (4 points) What is the output of the following code:

```
int a[] = {1,10,20}, *p=a, i=1, j=2;
for (i=0, p=a; i < 2; i++) {
  int j = *p;  p++;  *p = *p + j;
  printf("in: i=%d j=%d a=%d,%d,%d\n", i, j, a[0], a[1], a[2]);
}
printf("out: i=%d j=%d a=%d,%d,%d\n", i, j, a[0], a[1], a[2]);
```

Solution:

```
in: i=0 j=1 a=1,11,20       [1.3 points approx.]
in: i=1 j=11 a=1,11,31      [1.3 points approx.]
out: i=2 j=2 a=1,11,31      [1.3 points approx.]
```



Output:

in: i=0 j=1 a=1,11,20
in: i=1 j=11 a=1,11,31
out: i=2 j=2 a=1,11,31

b) (4 points) Write a C function `sort2` which can be used to do a "mini-sort" of two integer variables, by swapping their values only if the first variable is larger than the second. For example, after executing the following code: `int a=78, b=51; sort2(&a, &b);` the values of the variables would be `a=51 b=78`, but if we execute `sort2(&a, &b);` again, the values would not be changed.

Solution:

```
void sort2(int *pa, int *pb) {
  if ( *pa > *pb ) {
    int t = *pa; *pa = *pb; *pb = t;
  }
}
```

5. (10 points) C Program.

We will call a sequence of integers a *slow-changing sequence* if difference between any two consecutive numbers in sequence is at most 1. Write a C program that reads a positive integer n and prints all slow-changing sequences of non-negative integers that start with 0 and have length n. For example, for $n = 3$, the program should print sequences: `0 0 0`, `0 0 1`, `0 1 0`, `0 1 1`, and `0 1 2`. You do not need to check for errors in input.

(5 point option): For a partial solution of 5 points, write a function that checks whether an array is a slow-changing sequence.

Solution:

```
#include <stdio.h>

void f(int k, int n, int a[n]);

int main() {
  int n;
  scanf("%d", &n);
  int a[n];
  a[0] = 0;
  f(1, n, a);
  return 0;
}

void f(int k, int n, int a[n]) {
  int i;
  if (k == n) {
    int i;
    for (i=0; i < n; i++)
      printf(" %d", a[i]);
    printf("\n");
    return;
  }
  if (a[k-1] > 0) {
    a[k] = a[k-1] - 1;
    f(k+1, n, a);
  }
  a[k] = a[k-1];
  f(k+1, n, a);
  a[k] = a[k-1] + 1;
  f(k+1, n, a);
}
```

A 5-point solution:

```
int check_slow(int n, int a[n]) {
  int i;
  for (i=0; i<n-1; i++) {
    if (a[i] - a[i+1] < -1) return 0;
    if (a[i] - a[i+1] > 1)  return 0;
  }
  return 0;
}
```

Another 5-point solution: If we want to check that the sequence is non-negative as well, that is also a valid solution:

```
int check_slow(int n, int a[n]) {
  int i;
  for (i=0; i<n-1; i++) {
    if (a[i] < 0 || a[i+1] < 0) return 0;
    if (a[i] - a[i+1] < -1) return 0;
    if (a[i] - a[i+1] > 1)  return 0;
  }
  return 0;
}
```