Exam 1

April 2nd, 2018

Work alone. Do not use any notes or books. You have approximately 75 minutes to complete this exam.

Please write your answers on the exam. More paper is available if you need it. Please put your name at the top of the first page.

There are four questions on this exam, for a total of 80 points.

1 Countdown (20 points)

Write a function ‘printEvensBackwards‘, declared as below, that takes a character array and its size as an argument, and prints the characters at even-numbered positions in the array in reverse order. For example, calling `printEvensBackwards(10, "0123456789")` would print `86420`.

The caller guarantees that `n` will match the size of `a`, but there are no other constraints on `n` and `a`. You may assume that any necessary header files have already been included for you.

```
void printEvensBackwards(size_t n, const char *a);
```

Solution

```
void printEvensBackwards(size_t n, const char *a)
{
    // catch the empty string to avoid subtracting from an unsigned value
    if(n > 0) {
        // reduce n to the first even number less than n
        n = n - 1;
        n = n - (n%2);
        // print out the even numbers greater than 0
        while(n > 0) {
            putchar(a[n]);
            n -= 2;
        }
        // print out a[0]
        putchar(a[0]);
    }
}
```

2 String processing (20 points)

What output does the following program produce? You may assume that it is running on a machine that uses the standard ASCII character encoding, which in particular encodes `'a'` through `'z'` in the usual alphabetic order as 97 through 122.

Please draw a rectangle around your answer to distinguish it from any notes you may make.

```
#include <stdio.h>
#include <string.h>
void shift(char *s, int n) { do { s[n]++; } while(n-- > 0); }
int main(int argc, char **argv) {
    int n = 2; char word[6]; strcpy(word, "hello");
    shift(word, n); shift(word, n); shift(word, 4);
    puts(word);
    return 0;
}
```

Solution

khomp

3 Taking turns (20 points)

What output does the following program produce?

Please draw a rectangle around your answer to distinguish it from any notes you may make.

```
#include <stdio.h>
struct thing { void (*t)(int *, int *, struct thing *); };
void g(int *, int *, struct thing *);
void f(int *x, int *y, struct thing *t) { y[*x]+=1; (*x)+=1; t->t = g; }
void g(int *x, int *y, struct thing *t) { y[*x]+=2; (*x)*=2; t->t = f; }
#define N (8)
int main(int argc, char **argv) {
    struct thing t; t.t = f; int a[N];
    for(int i = 0; i < N; i++) { a[i] = i; }
    for(int i = 0; i < N; i++) { t.t(&i, a, &t); }
    for(int i = 0; i < N; i++) { printf("%d ", a[i]); }
    return 0;
}
```

Solution

1 1 4 3 4 6 6 9

4 Reversing a list (20 points)

The following function is supposed to reverse a linked list. For example, if `list` holds values 0, 1, 2, then the returned list should hold values 2, 1, 0. The reversal is destructive: the new list is built from pieces of the old list, and the caller should not attempt to use the old list after calling this function.

However, the function doesn't do what it is supposed to do. Explain what goes wrong, and write a revised version that works correctly.

```
struct elt { struct elt *next; int value; };
struct elt *reverse(struct elt *list) {
    struct elt *newList = 0;
    while(list) {
        list->next = newList;
        newList = list;
    }
    return newList;
}
```

Solution

The main problem is that `list` never changes in the body of the while loop, the loop will never finish unless the list is empty to start with. Here is one possible fixed version.

```
struct elt { struct elt *next; int value; };
struct elt *reverse(struct elt *list) {
    struct elt *newList = 0;
    struct elt *next;
    while(list) {
        next = list->next;
        list->next = newList;
        newList = list;
        list = next;
    }
    return newList;
}
```