SOFTWARE – Ph.D. Qualifying Exam Fall 2018

(i) (6 pts.)

Consider the following C program, which consists of two function definitions including the **`main`** function.

```
#include <stdio.h>

void f1(int *x, int j, int k) {
    int v = (*x);
    if (k > 0) {
        printf(".");
    } else {
        k = 1;
    }
    printf("%d", v);
    if (v >= 0) {
        x++;
        f1(&(x[j]), j, k);
    }
}

int main(void) {
    int values[] = {2, -3, 7, 4, 2, 5, -6,
            -5, 8, 4, -7, -5, -2, -100};

    f1(values, 1, 0);
    printf("\n---\n");
    f1(values, 2, 0);
    printf("\n---\n");
    f1(values, 3, 0);
    return 0;
}
```

Show the complete output as it appears on standard output. Show all work, and clearly indicate your solution. Show your work and your solution for this problem *only* on *this* page and (if more space is needed) *the next* page.

In case of an illegal dereferencing of a pointer (e.g., dereferencing of an uninitialized pointer, null pointer, or pointer that goes beyond the boundaries of an array), show all of the output from the **`printf`** calls that are executed up to the point just before the illegal pointer dereference, and then write "illegal pointer operation" on the following line.

This page is reserved as extra space for working on and writing your solution to Question (i).

(ii) (4 pts.)

Consider the following C program, which consists of two function definitions including the **`main`** function.

```
#include <stdio.h>
#include <ctype.h>
#define SIZE (27)

int f(char *s) {
    int counts[SIZE];
    int i = 0, M = 0, N = 0;
    int val = 0, index = 0;
    char c = '\0';


    for (i = 0; i < SIZE; i++) {
        counts[i] = 0;
    }

    for (i = 0; s[i] != '\0'; i++) {
        c = s[i];
        if (islower(c)) {
            index = c - 'a';
            (counts[index])++;
        } else if (isupper(c)) {
            index = c - 'A';
            (counts[index])++;
        } else {
            (counts[SIZE - 1])++;
        }
    }

    for (i = 0; i < SIZE; i++) {
        val = counts[i];
        if (val > M) {
            M = val;
            N = 1;
        } else if (val == M) {
            N++;
        }
    }

    printf("M: %d, N: %d\n", M, N);

    return 0;
}

int main(void) {
    f("Hello, how are you?\n");
    f("Great!");
    f("Maryland Terrapins");
    f("Score: 122 - 108\n");
    return 0;
}
```

Problem 2 continued: Show the complete output as it appears on standard output. Show all work, and clearly indicate your solution. Show your work and your solution for this problem *only* on *this* page and (if more space is needed) *the previous* page.

In case of an illegal dereferencing of a pointer (e.g., dereferencing of an uninitialized pointer, null pointer, or pointer that goes beyond the boundaries of an array), show all of the output from the **`printf`** calls that are executed up to the point just before the illegal pointer dereference, and then write "illegal pointer operation" on the following line.

(iii) (6 pts.)

Consider the following C program, which consists of a struct declaration, and three function definitions, including the **`main`** function.

```
#include <stdio.h>
#include <stdlib.h>

struct elem {
    int val;
    struct elem *next;
};

void add_two(struct elem **h, int v1, int v2) {
    struct elem *t1 = NULL, *t2 = NULL;
    struct elem *start = NULL, *end = NULL;
    struct elem *p = NULL, *q = NULL;

    t1 = malloc(sizeof(struct elem));
    t2 = malloc(sizeof(struct elem));
    t1->val = v1;
    t2->val = v2;
    t1->next = NULL;
    t2->next = NULL;

    if (v1 > v2) {
        start = t2;
        end = t1;
    } else {
        start = t1;
        end = t2;
    }
    start->next = (*h);
    (*h) = start;

    for (p = start; p != 0; p = p->next) {
        q = p;
    }
    q->next = end;
}

void disp(struct elem *h) {
    int i = 0;
    struct elem *p = NULL;

    for (p = h; p != NULL; p = p->next) {
        if (p != h) {
            printf(", ");
        }
        printf("%d", p->val);
    }
    printf("\n");
}
```

```
int main(void) {
    struct elem *data = NULL;
    int values[] = {8, 4, 12, 3, 5, -5, 8, 7, 2, 1};
    const int count = 10;
    const int start = 4, offset = 2;
    int i = start;

    while ((i != start) || (data == NULL)) {
        add_two(&data, values[i], values[i + 1]);
        disp(data);
        i = ((i + offset) % count);
    }
    return 0;
}
```

Show the complete output as it appears on standard output. Show all work, and clearly indicate your solution. Show your work and your solution for this problem *only* on *this* page and (if more space is needed) *the previous* page.

In case of an illegal dereferencing of a pointer (e.g., dereferencing of an uninitialized pointer, null pointer, or pointer that goes beyond the boundaries of an array), show all of the output from the **`printf`** calls that are executed up to the point just before the illegal pointer dereference, and then write "illegal pointer operation" on the following line.

(iv) (4 pts.)

Write a function that takes a double value `x`, integer value `n`, and pointer `p` as arguments, and sets the value pointed to by p to be equal to the result of the following cosine power series computation.

$$1 - \frac{x^2}{2!} + \frac{x^4}{4!} - \frac{x^6}{6!} + \ldots + (-1)^n \frac{x^{2n}}{(2n)!}$$

The prototype of the function is as follows:

```
void cosine_function(double x, double *p, int n);
```

Your solution should contain no function calls — i.e., no calls to any standard C library functions (including functions from `math.h`) or to any user-defined functions.

No error checking is required in the function.

Develop a complete C code implementation of the function `cosine_function`.

Show all work, and clearly indicate your solution. Show your work and your solution for this problem *only* on this page and (if more space is needed) *the next* page

This page is reserved as extra space for working on and writing your solution to Question (iv).

Software Qualifying Exam Solutions

Fall 2018
Dept. of ECE, University of Maryland, College Park
5/31/2018

Problem 1:

```
2.7.2.-6
---
2.4.-6
---
2.2.8.-2
```

Problem 2:

```
M: 6, N: 1
M: 1, N: 6
M: 3, N: 2
M: 12, N: 1
```

Problem 3:

```
-5, 5
7, -5, 5, 8
1, 7, -5, 5, 8, 2
4, 1, 7, -5, 5, 8, 2, 8
3, 4, 1, 7, -5, 5, 8, 2, 8, 12
```

Problem 4:

```
void cosine_function(double x, int n, double *p) {
    double term = 1;
    double factor = 1;
    double factorial_update = 1;
    double result = 1;
    int i = 0;
    double square = x * x;

    for (i = 1; i <= n; i++) {
        factor *= (-1);
        factorial_update = (2 * i - 1) * (2 * i);
        term *= square / factorial_update;
        result += (term * factor);
    }

    (*p) = result;
}
```