SOFTWARE – Ph.D. Qualifying Exam Spring 2017

(i) (6 pts.)

Consider the following C program, which includes three function definitions, including the `main` function.

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

void g(char *x) {
    int length = strlen(x);
    char *p = NULL;
    if (length != 0) {
        printf("%c\n", x[0]);
        p = &(x[2]);
        g(p);
    }
}

void f(char a[], int x, int *y) {
    x = x + (*y);
    a[x] = '\0';
    a[x + 1] = '\0';
    (*y)++;
    g(a);
}

int main(void) {
    char s[] = "Maryland";
    int x1 = 4;
    int x2 = 2;

    printf("s = %s, x1 = %d, x2 = %d\n",
            s, x1, x2);
    f(s, x1, &x2);
    printf("s = %s, x1 = %d, x2 = %d\n",
            s, x1, x2);

    return 0;
}
```

Show the complete output as it appears on standard output. <u>Show all work</u>, and clearly indicate your solution. Show your work and your solution for this problem *<u>only</u>* 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.

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

void display_result(int result) {
    static int index = 1;
    printf("result #%d: %d\n", index++, result);
}

int main(void) {
    char *s = "123xyz456abc", *p = s;
    int x = 0, iterations = 0, i = 0;
    const int step = 3, lim1 = 20, lim2 = 63, param1 = 4;

    iterations = 0;
    for (i = 0; i <= lim1; i += step) {
        iterations++;
    }
    display_result(iterations);

    iterations = 0;
    x = lim2;
    while (x > 0) {
        x = x / param1;
        iterations++;
    }
    display_result(iterations);

    iterations = 0;
    for (i = 17; i > 3; i--) {
        i -= 2;
        iterations++;
    }
    display_result(iterations);

    iterations = 0;
    while (!(isalpha(*p))) {
        p++;
        iterations++;
    }
    display_result(iterations);

    iterations = 0;
    for (i = 10; (i + iterations) < 20; i++) {
        if ((i % 3) == 0) {
            iterations += 2;
        } else {
            iterations++;
        }
    }
    display_result(iterations);
    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.

(iii) (6 pts.)

Consider the following C program, which includes three function definitions, including the **main** function shown on the following page.

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

typedef struct elem {
    int data;
    struct elem *next;
    struct elem *prev;
} elem;

void insert(elem **h, elem *e) {
    e->next = (*h);
    (*h)->prev = e;
    (*h) = e;
    e->prev = NULL;
}

elem *get(elem *h, char *config) {
    elem *p = h;
    int length = strlen(config), i = 0;

    for (i = 0; i < length; i++) {
        if (p == NULL) {
            fprintf(stderr, "invalid access\n");
            exit(1);
        }
        if (config[i] == '+') {
            p = p->next;
        }
        else if (config[i] == '/') {
            p = p->prev;
        }
    }

    return p;
}
```

```
int main(void) {
    elem a = {5, NULL, NULL};
    elem *b = NULL; elem *head = &a;
    int data[] = {1, 2, 3, 4, 5, 6, 7, 8, 9, 0};
    int i = 0;
    const int len = 10, factor = 7, insertions = 6;

    for (i = 0; i < insertions; i++) {
        b = malloc(sizeof(elem));
        b->data = data[(i * factor) % len];
        printf("inserting: %d\n", b->data);
        insert(&head, b);
    }
    b = get(head, "+aa/++");
    printf("<%d>\n", b->data);
    b = get(b, "/x+y/");
    printf("<%d>\n", b->data);
    b = get(b, "zz++/+/z++");
    printf("<%d>\n", b->data);
    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.)

Consider the following string of **`if-else`** statements

```
if (c == '+') {
    result = x1 + x2;
} else if (c == '-') {
    result = x1 - x2;
} else if (c == '*') {
    result = x1 * x2;
} else {
    result = 0;
}
```

Write a new version of this code segment that has the same functionality but uses a **`switch`** statement instead of **`if-else`** for the required program selection operations.

Show all work, and clearly indicate your solution. Show your work and your solution for this problem *only* on the *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

Spring 2017
Dept. of ECE, University of Maryland, College Park
11/20/2016

Problem 1:

```
s = Maryland, x1 = 4, x2 = 2
M
r
l
s = Maryla, x1 = 4, x2 = 3
```

Problem 2:

```
result #1: 7
result #2: 3
result #3: 5
result #4: 3
result #5: 6
```

Problem 3:

```
inserting: 1
inserting: 8
inserting: 5
inserting: 2
inserting: 9
inserting: 6
<2>
<9>
<8>
```

Problem 4:

```
switch (c) {
    case '+':
        result = x1 + x2;
        break;
    case '-':
        result = x1 - x2;
        break;
    case '*':
        result = x1 * x2;
        break;
    default:
        result = 0;
        break;
}
```