SOFTWARE – Ph.D. Qualifying Exam Spring 2018

(i) (5 pts.)

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

```
#include <stdio.h>

int g(int z) {
    int y = 0;
    const int k = 3;

    printf("z = %d\n", z);

    if (z <= 0) {
        return k * (z + 2);
    } else {
        y = g(z - 1) + g(z - 2);
        return y;
    }
}

int main(void) {
    int arg = 3;
    int result = 0;

    result = g(arg);

    printf("result = %d\n", result);

    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) (6 pts.)

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

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

#define LEN 9
#define OFFSET 5

typedef struct node {
    int val;
    struct node *next;
} node;

void insert_one(node **h, node *e) {
    int v = e->val;
    node *p = NULL;

    if (v < -1) {
        e->next = (*h)->next->next;
        (*h)->next->next = e;
    } else if (v > 1) {
        e->next = (*h);
        (*h) = e;
    } else {
        for (p = (*h); p->next != NULL; p = p->next);
        e->next = p->next;
        p->next = e;
    }
}

void show_all(node *h) {
    node *p = NULL;

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

```
int main(void) {
    node n1 = {7, NULL};
    node n2 = {3, &n1};
    node n3 = {5, &n2};
    node *h = &n3;
    int values[] = {7, -4, 9, -1, 0, 0, 7, -8, 12};
    int i = 0;
    node *new = NULL;

    show_all(h);
    show_all(&n1);

    for (i = 0; i < LEN; i+=2) {
        new = malloc(sizeof(node));
        new->val = values[(i + OFFSET) % LEN];
        insert_one(&h, new);
    }

    show_all(h);
    show_all(&n2);
}
```

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) (5 pts.)

Consider the following C program.

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

#define SIZE 10

int main(void) {
    char src1[] = "xya$%fsp\\my";
    char src2[] = "\\m%psdf22nnz";
    char dst[SIZE] = "123456789";
    int i = 0;
    const int stride = 2;

    for (i = 0; i < SIZE; i += stride) {
        if (isalpha(src1[i])) {
            dst[i] = src2[i];
        } else {
            dst[i] = src1[i];
            src1[i] = '\0';
        }
    }

    printf("%s\n", dst);
    printf("len1 = %d\n", (int)(strlen(src1)));
    printf("len2 = %d\n", (int)(strlen(src2)));

    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 (iii).

(iv) (4 pts.)

For this problem, we define the set of *valid* text characters as the set that consists of (1) all alphabetic characters (lower case and upper case), (2) all characters that correspond to decimal digits (0-9), (3) the space character, and (4) the underscore character. Write a C program that inputs a line of text from standard input, and prints the line to standard output with all of the non-valid text characters filtered out (removed). For example, if the input line is

```
Hello ^_^ How Are You?
```

then the output would be

```
Hello _ How Are You
```

Write the program to handle an input line that has at most 100 characters. If the input line has more than 100 characters, then print a meaningful error message to standard error and exit without printing anything to standard output. Use the **`putc`** utility to read the input text from standard input.

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

Spring 2018
Dept. of ECE, University of Maryland, College Park
11/20/2018

Problem 1:

```
z = 3
z = 2
z = 1
z = 0
z = -1
z = 0
z = 1
z = 0
z = -1
result = 24
```

Problem 2:

```
<5><3><7>
<7>
<9><7><5><3><-8><7><0><0>
<3><-8><7><0><0>
```

Problem 3:

```
\2%4%6f8\
len1 = 4
len2 = 12
```

Problem 4:

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

#define MAXLEN 100

int main(void) {
    char line[MAXLEN + 1];
    int i = -1;
    char c = '\0';
    int done = 0;

    while (!done) {
        c = getc(stdin);
        if ((c == EOF) || (c == '\n')) {
            done = 1;
        } else {
            i++;
            if (i >= MAXLEN) {
                fprintf(stderr, "Line too long\n");
                exit(1);
            } else {
                line[i] = c;
            }
        }
    }
    line[++i] = '\0';

    for (i = 0; i < strlen(line); i++) {
        c = line[i];
        if (isalpha(c) || isdigit(c) ||
                (c == ' ') || (c == '_')) {
            printf("%c", c);
        }
    }
    printf("\n");
    return 0;
}
```