

CSE 333 Midterm Exam 7/28/14

Name _____

There are 5 questions worth a total of 100 points. Please budget your time so you get to all of the questions. Keep your answers brief and to the point.

The exam is closed book, closed notes, closed electronics, closed telepathy, open mind.

If you don't remember the exact syntax for something, make the best attempt you can. We will make allowances when grading.

Don't be alarmed if there seems to be more space than is needed for your answers – we tried to include more than enough blank space.

Relax, you are here to learn.

Please wait to turn the page until everyone is told to begin.

Score _____ / 100

1. _____ / 20

2. _____ / 12

3. _____ / 24

4. _____ / 24

5. _____ / 20

CSE 333 Midterm Exam 7/28/14

Question 1. (20 points) C programming. For this question implement a C function `contains` that returns 1 (true) if a given C string appears as a substring of another C string starting at a given position.

Examples: The following should return 1 (true): `contains("abc", "abc", 0)` ;
`contains("bc", "abc", 1)` ; `contains("xyz", "xyzzxy", 0)`. The
following calls return 0 (false): `contains("abc", "xyz", 1)` ;
`contains("abc", "abcde", 1)` ; `contains("def", "abcde", 3)` ;
`contains("de", "abcde", 1)`.

Restriction: the only function you may use from the `<string.h>` library is `strlen`.

Simplifying assumptions: Assume that pointer arguments are non-null and point to properly `'\0'`-terminated C strings that have at least one character in them besides the terminating `'\0'`. You may also assume that `pos` is non-negative (i.e, `pos >= 0`).

```
// return 1 if string s appears in target starting at pos
int contains (char *s, char *target, int pos) {
```

```
}
```

CSE 333 Midterm Exam 7/28/14

Question 2. (12 points) Here is a small header file and a main program that includes it.

File `midterm.h`

```
#ifndef _MIDTERM_H_
#define _MIDTERM_H_

#define NEW_PI PI+1
#define GOLDEN 1.618
#define NEWER_PI(x) (x*PI)

typedef int number;

#endif // _MIDTERM_H_
```

File `midterm.c`

```
#include "midterm.h"

#define PI 3.1415
#define GOLDEN 1.618+1

int main(int argc, char **argv) {
    number x = PI;
    int xx = NEW_PI;
    int y = GOLDEN;
    int z = NEWER_PI(y+1);
    return 0;
}
```

Below, write the output produced by the C preprocessor (`cpp` or `gcc -E`) showing exactly how file `midterm.c` is rewritten and sent to the compiler phase of `gcc`. (The preprocessor can correctly process this file without any blocking errors.)

CSE 333 Midterm Exam 7/28/14

Question 3. (24 points) The somewhat traditional, always annoying C program. Consider the following program, which does compile and execute without any errors. (We assume for this question that `malloc` always succeeds and returns a non-null pointer.)

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

typedef struct point2d { // 2-d point
    int x, y;
} Point2d, *Point2dPtr;

void PrPoint(char *label, Point2dPtr p) {
    printf("%s %d %d\n", label, p->x, p->y);
}

Point2dPtr NewPt(int a, int b) {
    Point2dPtr p = (Point2dPtr)malloc(sizeof(Point2d));
    p->x = a;
    p->y = b;
    return p;
}

void Shuffle(Point2d a, Point2dPtr x, Point2dPtr y, Point2dPtr *z) {
    *z = y;
    *x = *y;
    **z = a;
    //// HERE ////
}

int main() {
    Point2dPtr p1 = NewPt(1, 2);
    Point2dPtr p2 = NewPt(3, 4);
    Point2dPtr p3;
    Shuffle(*p1, p1, p2, &p3);
    PrPoint("p1", p1);
    PrPoint("p2", p2);
    PrPoint("p3", p3);
    return 0;
}
```

Answer questions about this program on the next page. You may remove this page for reference if you wish.

CSE 333 Midterm Exam 7/28/14

Question 3. (cont.) (a) (15 points) Draw a boxes ‘n arrows diagram showing state of memory when control reaches the comment containing `//// HERE ////` in the middle of function `Shuffle`. Your diagram should have boxes showing the stack frames for all active functions. The stack frames should show values of all local variables. Draw each `point2d struct` as a box with two labeled fields `x` and `y`. Draw an arrow from each pointer to the location that it references. Data that is allocated on the heap should be drawn in a separate area, since it is not part of any function stack frame After drawing your diagram, be sure to answer part (b) at the bottom of the page.

(b) (9 points) What output does this program produce when it is executed?

CSE 333 Midterm Exam 7/28/14

Question 4. (24 points) Here is an attempt to fix the buggy vector program. Be sure to read the rest of the question before tearing out any pages or doing any work. ...

```
1 #include <stdbool.h>          // for bool
2 #include <stdlib.h>          // for malloc, free
3
4 #define INIT_VEC_SIZE 8
5 #define FINAL_VEC_SIZE 10
6
7 typedef void* element_t;
8
9 typedef struct vector_t {
10     size_t length;          // length of array
11     element_t *array;      // data items - unused entries are NULL
12 } *vector_t;
13
14 // Fill the array in v with the elements 0 to n-1
15 // Return true if it succeeds
16 bool FillElements(size_t n, vector_t v);
17
18 // Create a vector and then add some values to the vector array
19 int main(int argc, char *argv[]) {
20
21     // allocate initial vector and exit if failure
22     vector_t v = (vector_t)malloc(sizeof(struct vector_t));
23     if (v == NULL)
24         return EXIT_FAILURE;
25     v->array = (element_t*)malloc(INIT_VEC_SIZE *sizeof(element_t));
26     if (v->array == NULL) {
27         free(v);
28         return EXIT_FAILURE;
29     }
30
31     // save length in vector struct and initialize array elements
32     v->length = INIT_VEC_SIZE;
33     bool success = FillElements(FINAL_VEC_SIZE, v);
34     if(!success)
35         return EXIT_FAILURE;
36 }
37
38 // fill the array in vwith the elements 0 to n-1
39 // return true if successful
40 bool FillElements(size_t n, vector_t v){
41     size_t i;
42
43     // store new int objects in so vector[i] holds int i.
44     // exit if any step fails.
45     for (i = 0; i < FINAL_VEC_SIZE; ++i) {
46         int *x = (int*)malloc(sizeof(int));
47         if (x == NULL)
48             return false;
49         *x = i;
50         v->array[i] = x;
51     }
52     return true;
53 }
```

CSE 333 Midterm Exam 7/28/14

Question 4. (cont.) There seem to be some memory problems, so we ran this program using valgrind. Here's the output we got:

```
==9424== Memcheck, a memory error detector
==9424== Copyright (C) 2002-2012, and GNU GPL'd, by Julian Seward et al.
==9424== Using Valgrind-3.8.1 and LibVEX; rerun with -h for copyright info
==9424== Command: ./buggy
==9424==
==9424== Invalid write of size 8
==9424==   at 0x400671: FillElements (buggy.c:50)
==9424==   by 0x4005FB: main (buggy.c:33)
==9424== Address 0x4c2a0d0 is 0 bytes after a block of size 64 alloc'd
==9424==   at 0x4A06409: malloc (in /usr/lib64/valgrind/vgpreload_memcheck-amd64-linux.so)
==9424==   by 0x4005B4: main (buggy.c:25)
==9424==
==9424==
==9424== HEAP SUMMARY:
==9424==   in use at exit: 120 bytes in 12 blocks
==9424== total heap usage: 12 allocs, 0 frees, 120 bytes allocated
==9424==
==9424== 8 bytes in 2 blocks are definitely lost in loss record 1 of 4
==9424==   at 0x4A06409: malloc (in /usr/lib64/valgrind/vgpreload_memcheck-amd64-linux.so)
==9424==   by 0x40063B: FillElements (buggy.c:46)
==9424==   by 0x4005FB: main (buggy.c:33)
==9424==
==9424== 112 (16 direct, 96 indirect) bytes in 1 blocks are definitely lost in loss record 4 of 4
==9424==   at 0x4A06409: malloc (in /usr/lib64/valgrind/vgpreload_memcheck-amd64-linux.so)
==9424==   by 0x400598: main (buggy.c:22)
==9424==
==9424== LEAK SUMMARY:
==9424==   definitely lost: 24 bytes in 3 blocks
==9424==   indirectly lost: 96 bytes in 9 blocks
==9424==   possibly lost: 0 bytes in 0 blocks
==9424==   still reachable: 0 bytes in 0 blocks
==9424==   suppressed: 0 bytes in 0 blocks
==9424==
==9424== For counts of detected and suppressed errors, rerun with: -v
==9424== ERROR SUMMARY: 4 errors from 3 contexts (suppressed: 2 from 2)
```

Your job is to find all of the bugs in the program and indicate how to fix them. Use the information in the program itself as well as the valgrind output to locate the bugs (and it may be that there are some bugs that don't result in any error report from valgrind, so be sure to check the source code carefully).

You can either mark the original program to show what's wrong and how to fix it, or you can write an explanation on the following page. Feel free to detach this page (with the valgrind output) while you work, but be sure to turn in the page with the program code if you mark your answers there.

CSE 333 Midterm Exam 7/28/14

Question 4. (cont.) You can use the space on this page for your answer if you wish, or you can mark your answers on the program code. Just be sure the grader can follow what you've done to give you as much credit as possible.

CSE 333 Midterm Exam 7/28/14

Question 5. (20 points) A little C++ programming. We've been having so much fun with linked lists, it seemed a shame not to do a bit more in C++. Here is the header for a class that implements a list of C++ strings using a single-linked list made up from StringNode objects.

```
#include <string>
using namespace std;

class StringNode {    // single node in a linked list of strings
public:
    string value_;    // value in this node
    StringNode * next_; // next node in the list or nullptr if none

    // convenience constructor - initialize new node with string s
    // and next pointer p
    StringNode(string s, StringNode *p): value_(s), next_(p) { }
};

class StringLinkedList {
public:

    // construct empty StringLinkedList
    StringLinkedList() : first_(nullptr), last_(nullptr) { }

    // additional operations omitted...

private:
    StringNode * first_;    // first node in list
    StringNode * last_;    // last node in list
                        // first_ = last_ = nullptr if the
                        // list is empty.
};
```

You should assume that all dynamic memory allocation and deletion functions will succeed without errors – you do not need to check for failures or handle exceptions.

You can remove this page for reference while working on the question if you wish.

CSE 333 Midterm Exam 7/28/14

Question 5. (cont.) We would like to add a += assignment operator to this class to append new strings to a list. If `lst` is an object of type `StringLinkedList`, then the operation `lst+=s` should create a new node holding the C++ string `s` and add it (append it) as the last node of the list `lst`.

(a) (5 points) Give the declaration for this new += operation to be added to the declaration of class `StringLinkedList` (above).

(b) (15 points) Give the full implementation of this added += operation as it would appear in a separate source file `StringLinkedList.cc`. Your answer should contain any necessary `#include` or other compiler and preprocessor directives needed.