

CSE 333 Midterm Exam 2/12/16

Name _____ UW ID# _____

There are 6 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. _____ / 12

2. _____ / 24

3. _____ / 22

4. _____ / 22

5. _____ / 15

6. _____ / 5

CSE 333 Midterm Exam 2/12/16

Question 1. (12 points) Preprocessor. Consider the following C (*not* C++ files).

```
=====
func.h
=====
#ifndef _FUNC_H_
#define _FUNC_H_
#define FUN(a,b) a*b
#endif

=====
nums.h
=====
#ifndef _NUMS_H_
#define _NUMS_H_
#ifdef BIG
typedef long int num;
#else
typedef int num;
#endif
#define NBR 3
#endif

=====
test.c
=====
#include <stdio.h>
#include "nums.h"
#include "func.h"

#define BIG

num compute(int x) {
    return FUN(x+1,NBR);
}

int main() {
    printf("%d\n", compute(2));
    return 0;
}
```

(a) (10 points) Give the output produced by the preprocessor (`cpp -P test.c`) when it reads and processes the file `test.c`. Ignore the `#include <stdio.h>` line – it will insert the declarations from `stdio.h` and do nothing further. Otherwise, your answer should show all of the output from the preprocessor. There are no preprocessor errors in this program, and the resulting program compiles and executes without errors.

(b) (2 points) What does this program print when it is compiled and executed?

CSE 333 Midterm Exam 2/12/16

Question 2. (24 points) C programming – with HashTables this time! This question involves the data structures from the HW1 and HW2 projects. Copies of the `LinkedList.h`, `HashTable.h`, and `HashTable_priv.h` header files have been provided on separate pages.

For this problem give the implementation of a new function `HashTableValues` to be added to `HashTable.c`. This function should return a newly-allocated array containing copies of the values stored in the `HashTable` and also return the number of values in (i.e., the size of) the new array. The function result should be 1 if it is able to successfully allocate an array of the proper size and fill it with copies of the values found in the `HashTable` (just the values, not the keys or `<key,value>` pairs). The function result should be 0 if some error occurs. The array might, of course, wind up containing some duplicate values if the same value occurs more than once in the `HashTable` in separate `<key,value>` pairs. The order of the values is not specified since the elements of a `HashTable` are not ordered.

Here is some sample code that shows how this function could be used to retrieve the values in a `HashTable` and process them:

```
HTValue_t *values;
HWSize_t  nvalues;

// get value array and number of values
int res = HashTableValues(ht, &values, &nvalues);
Verify333(res == 1);

// use returned values
for (HWSize_t i = 0; i < nvalues; ++i) {
    HTValue_t val = values[i];
    // do something with val...
}

// free array when done
free(values);
```

Your answer may use any of the functions or data declared in the `LinkedList.h`, `HashTable.h`, and `HashTable_priv.h` headers. Don't be alarmed if the solution turns out to be fairly short.

Write your answer on the next page. You may remove this page from the exam while you are working on the question, but please return it at the end of the hour.

CSE 333 Midterm Exam 2/12/16

Question 2. (cont.) Write an implementation of function `HashTableValues`, below.

```
// Store in parameter nvalues the number of values in
// HashTable ht, and store in parameter values a pointer
// to a newly allocated array containing those values.
//
// Return 1 if the function is successful.
// Return 0 if some error occurs.

int HashTableValues(HashTable ht,
                   HTValue_t **values, HWSize_t *nvalues){
```

```
}
```

CSE 333 Midterm Exam 2/12/16

Question 3. (22 points) Pointy things. Consider the following program, which, in the customary manner, compiles and executes with no warnings or errors:

```
#include <stdio.h>

void swap(int **a, int **b) {
    printf("swap1: **a = %d, **b = %d\n", **a, **b);
    int *tmp = *a; *b = *a; *a = tmp;
    //HERE!!!
    printf("swap2: **a = %d, **b = %d\n", **a, **b);
}

void mumble(int **p, int *q) {
    swap(p, &q);
    **p = *q;
}

int main() {
    int k = 3;
    int n = 7;
    int *p = &k;
    mumble(&p, &n);
    printf("main: k = %d, n = %d, *p = %d\n", k, n, *p);
    return 0;
}
```

(a) (14 points) Draw a boxes 'n arrows diagram showing the memory layout and contents at the point just before the *second* printf in function swap is executed (marked with HERE!!! in the comment). Be sure your diagram clearly shows the values of all variables in all active functions and has a separate box (i.e., stack frame) for each active function. For each pointer, draw an arrow from the pointer to the variable that it references. Use the space below the code and/or to the right for your diagram.

(b) (8 points) What does this program print when it is executed?

CSE 333 Midterm Exam 2/12/16

Question 4. (22 points) The program on this page and the next opens two files, one for reading and one for writing, and copies the contents of the first file to the second. Your job is to complete the code by filling in the blanks lines with the correct POSIX I/O function calls to handle the files (open, close, read, write).

Here is a summary of some key POSIX I/O functions for your reference.

```
int open(const char *name, int mode);
    mode is one of O_RDONLY, O_WRONLY, O_RDWR
int creat(const char *name, int mode);
    create a new file
int close(int fd);
ssize_t read(int fd, void *buffer, size_t count);
    returns # bytes read or 0 (eof) or -1 (error)
ssize_t write(int fd, void *buffer, size_t count);
    returns # bytes written or -1 (error)
```

Below is the code you are to complete. You should assume that all necessary header files have been #included and you do not need write any other #includes.

```
#define SIZE 1024
int main(int argc, char** argv) {
    int fd1, fd2;
    char buf[SIZE];
    ssize_t rlen, total, wlen;

    if (argc != 3) {
        fprintf(stderr, "Usage: ./a.out <file1> <file2>\n");
        exit(1);
    }
    // open first file for reading
    _____;

    if (fd1 == -1) {
        fprintf(stderr, "Could not open file for reading\n");
        exit(1);
    }
    // create the second file
    fd2 = creat(argv[2], 0777);
    if (fd2 == -1) {
        close(fd1);
        fprintf(stderr, "Could not create file for writing\n");
        exit(1);
    }
}
```

(code continued on next page)

CSE 333 Midterm Exam 2/12/16

Question 4. (cont.) Continued from previous page.

```
// Copy all data from fd1 to fd2
do {
    // read next data from fd1 into buf
    _____;

    if (rlen == -1) {
        if (errno != EINTR) {
            close(fd1);
            close(fd2);
            perror(NULL);
            exit(1);
        }
        continue;
    }
    // Write newly read data from buf to fd2
    total = 0;

    while ( _____ ) {
        _____;

        if (wlen == -1) {
            if (errno != EINTR) {
                close(fd1);
                close(fd2);
                perror(NULL);
                exit(1);
            }
            continue;
        }
        _____;
    }
} while ( _____ );

// Close input and output files
close(fd1);
close(fd2);
return 0;
}
```

CSE 333 Midterm Exam 2/12/16

A few short-answer questions to finish up.

Question 5. (15 points) Here are three C functions that are supposed to return a pointer to a new C string value (null-terminated array of characters) that the caller is responsible for freeing when the caller is done with it. For each function, if it behaves as specified by the comment, say so. If there are one or more bugs in the code, explain what's wrong and show how to fix the function so it works properly.

(a)

```
// return a new string with a copy of str
char *clone(char *str) {
    char *result = (char *)malloc(sizeof(str)+1);
    strcpy(result, str);
    return result;
}
```

(b)

```
// return a new string containing "hello"
char *hello() {
    char *result = (char *)malloc(5);
    result = "hello";
    return result;
}
```

(c)

```
// return a new string containing "cse333"
char *cse333() {
    char result[] = "cse333";
    return result;
}
```


CSE 333 Midterm Exam 2/12/16

Question 6. (5 points) In one of the C++ exercises we defined assignment for Vectors. The function heading for assignment specified that it returned a reference:

```
Vector &Vector::operator=(const Vector &rhs) {
    if (this != &rhs) {
        x_ = rhs.x_;
        y_ = rhs.y_;
        z_ = rhs.z_;
    }
    return *this;
}
```

Suppose we accidentally omitted the & and wrote the function heading like this:

```
Vector Vector::operator=(const Vector &rhs) { ... }
```

Now assume we have three vectors, v1, v2, and v3 and we write the following chained assignment:

```
v1 = v2 = v3;
```

Amazingly enough everything still compiles and executes without crashing (although maybe not with exactly the right behavior). But something must be different. Describe exactly what's different when the chained assignment statement `v1=v2=v3;` is executed using the second definition of `operator=` compared to the first one.