

## CSE 333 Midterm Exam 7/27/15

**Question 5.** (20 points) Pointy things. Consider the following program, which compiles and executes with no warnings or errors:

```
#include <stdio.h>

void g(int **x, int *y, int *z) {
    **x = 10;
    *x = z;
    // HERE!!! (see below) //
    printf("g: %d %d %d\n", **x, *y, *z);
}

void f(int *q, int n, int *p) {
    n = n+2;
    *p = *p**q;
    g(&q, &n, p);
    printf("f: %d %d %d\n", *p, *q, n);
}

int main() {
    int a = 7;
    int b = 2;
    f(&a, b, &b);
    printf("main: %d %d\n", a, b);
    return 0;
}
```

(a) (14 points) Draw a boxes 'n arrows diagram showing the memory layout and contents at the point just before the `printf` in function `g` 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 or stack frame for each 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) (6 points) What does this program print when it is executed?



## 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 7/25/16

**Question 6.** (20 points) Not quite the traditional what-does-it-print question. Consider the following C++ program, which, as is usual, compiles and executes with no errors.

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

int f(int &n, int *pa, int &k, int *pb) {
    k = pb[1];
    pb[2] = pa[1];
    n = *pb**pa;
    return k+1;
}

int main() {
    int a = 1;
    int &b = a;
    int ray[4] = { 10, 11, 12, 13 };
    int *p = ray;
    int *q = &ray[1];

    *p = f(ray[2], p, b, q);

    cout<< "a = " << a << ", b = " << b << ", *q = " << *q << endl;
    cout<< "ray = ";
    for (int k = 0; k < 4; k++)
        cout << ray[k] << " ";
    cout << endl;

    return 0;
}
```

What output does this program produce when it runs? (You are not required to draw a boxes-n-arrows diagram, but you might find doing so to be very helpful, and it might help us if we need to assign partial credit to a not-completely-perfect answer.)

## CSE 333 18su Midterm Exam July 23, 2018

**Question 5.** (26 points) One of the summer interns is trying to learn C++ and has written the following class that stores an array of doubles and a main program that uses it.

```
class Doubles {
public:
    // construct Doubles given array and # elements
    Doubles(double *vals, uint32_t size)
        : v_(new double[size]), sz_(size) {
        for (uint32_t k = 0; k < size; k++)
            v_[k] = vals[k];
    }

    // destructor, other standard operations
    ~Doubles() { delete[] v_; }
    Doubles(const Doubles &other) = default;
    Doubles &operator=(const Doubles &other) = default;

    // "getter" functions
    double get(uint32_t k) const { return v_[k]; }
    uint32_t length() const { return sz_; }

private:
    double* v_; // heap-allocated array
    uint32_t sz_; // size of array
};

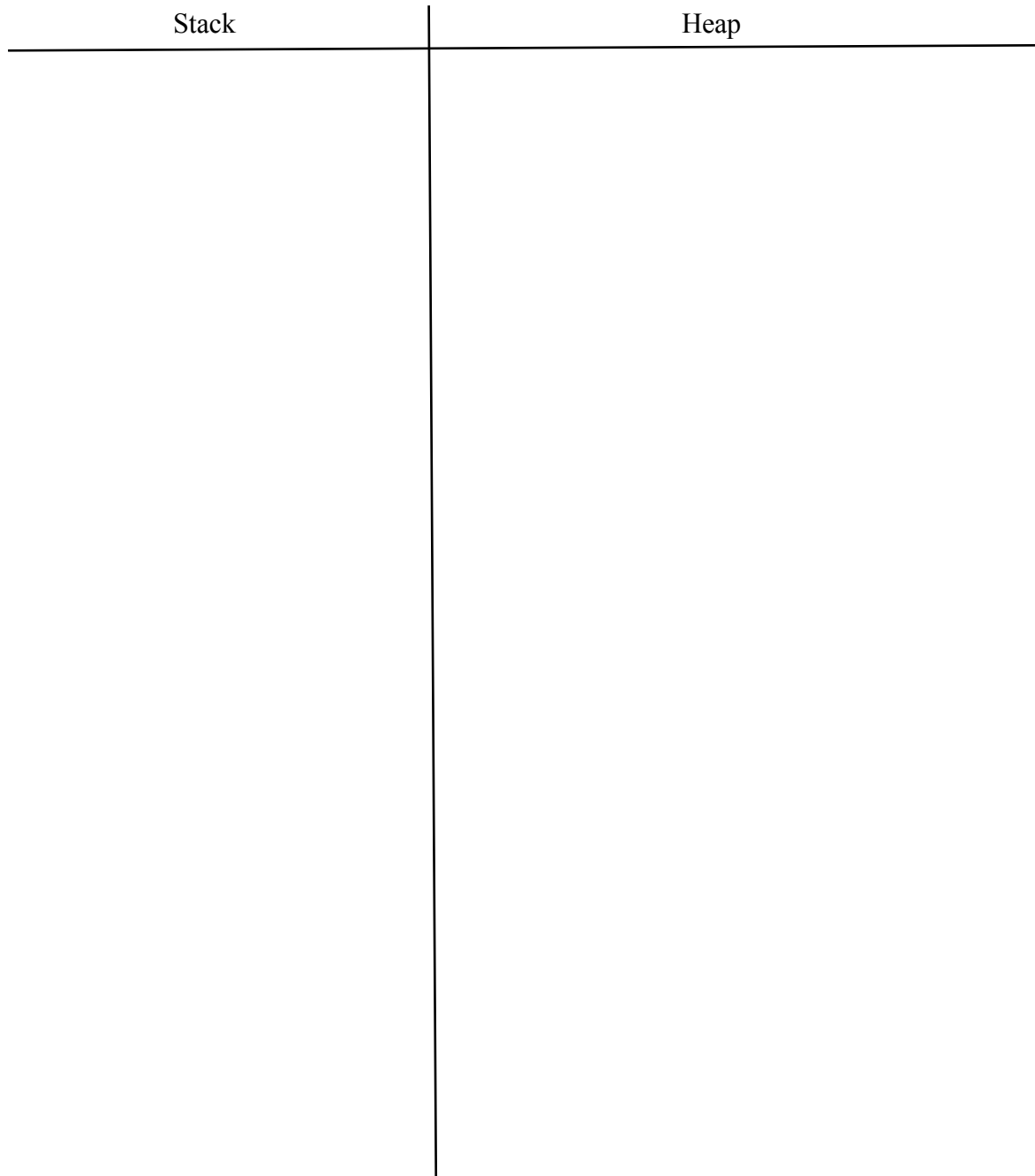
// print data in a Doubles object
void prdbl(Doubles d) {
    ///// ***>>> here <<<<*** /////
    cout << "[ ";
    for (uint32_t k = 0; k < d.length(); k++)
        cout << d.get(k) << " ";
    cout << "]" << endl;
}

int main() {
    double a[] = { 1.1, 2.2, 3.3 };
    Doubles d3(a, 3);
    Doubles* dp = new Doubles(d3);
    prdbl(d3);
    prdbl(*dp);
    delete dp;
    return 0;
}
```

Please answer the questions about this class on the next page and **remove this page from the exam. This page will not be scanned for grading.**

**CSE 333 18su Midterm Exam July 23, 2018**

**Question 5. (cont.)** (a) (12 points) Draw a *precise* diagram showing the contents of memory the **first time** execution reaches the comment `//// **>>>> here <<<<*** ////` at the beginning of function `prdbl`. Your diagram should clearly show the contents of the individual stack frames for `main` and `prdbl` and the contents of heap storage, with appropriate arrows from pointers to values that they reference. Then continue with the question on the next page.



(continued on next page)

## CSE 333 18su Midterm Exam July 23, 2018

**Question 5. (cont.)** (b) (3 points) When the program is executed it crashes. Exactly where does it crash, when, and why? (what is the problem?) (Be brief but precise!)

(c) (3 points) Our summer intern has been googling and thinks that something called the “Rule of 3” is the reason for the crash. The intern proposes replacing the destructor with the following code to match the copy constructor and assignment:

```
~Doubles() = default;
```

Will the program run without crashing if this is done? Why or why not? (briefly)

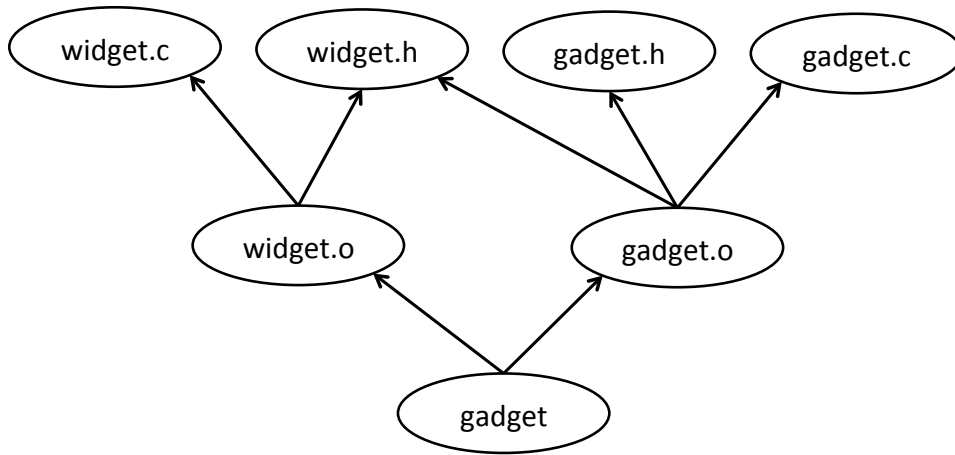
## **CSE 333 18su Midterm Exam July 23, 2018**

**Question 5. (cont.)** (d) (8 points) What really needs to be done to fix this class so it works properly and behaves appropriately for a C++ class? Give the changes needed below by listing which functions (methods) need to be changed in the original code and writing the correct code below.



## CSE 333 Midterm Exam 7/25/16

Make and related things. We have the following diagram that describes the dependencies between the various files that are involved in building an executable program named gadget.



**Question 2.** (9 points, 3 each) What can we conclude from the above diagram? In the statements below, **circle** the number of the *best* choice in each group of possible answers:

(a) The `widget.c` source file:

- (i) **Definitely** contains a `#include "widget.h"` preprocessor directive.
- (ii) **Might** contain a `#include "widget.h"` preprocessor directive, but might not.
- (iii) **Does not** contains a `#include "widget.h"` preprocessor directive.

(b) The `gadget.c` source file:

- (i) **Definitely** contains a `#include "widget.h"` preprocessor directive.
- (ii) **Might** contain a `#include "widget.h"` preprocessor directive, but might not.
- (iii) **Does not** contains a `#include "widget.h"` preprocessor directive.

(c) The `main` function for this program:

- (i) **Definitely** is implemented (defined) in the `gadget.c` source file.
- (ii) **Might** be implemented (defined) in the `gadget.c` source file, but might not.
- (iii) **Is not** implemented (defined) in the `gadget.c` source file.

(more about this diagram on the next page)

## 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 18au Midterm Exam Nov. 2, 2018

**Question 5.** (16 points) Constructor madness. Consider the following C++ program which does compile and execute successfully. On the next page, write the output produced when it is executed.

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

static int idnum = 1;    // global var: next obj id number

class obj {
public:
    obj() {                // default constructor
        id_ = idnum; idnum++;
        cout << "obj " << id_ << ": default constructor" << endl;
    }
    obj(int n) {           // int constructor
        id_ = idnum; idnum++;
        cout << "obj " << id_ << ": int constructor" << endl;
    }
    obj(const obj & other) { // copy constructor
        id_ = idnum; idnum++;
        cout << "obj " << id_ << ": copy constructor from " <<
            other.id_ << endl;
    }
    obj& operator=(const obj & other) { // assignment operator
        cout << "obj " << id_ << ": assignment operator from " <<
            other.id_ << endl;

        return *this;
    }
    ~obj() { // destructor
        cout << "obj " << id_ << ": destructor" << endl;
    }
private:
    int id_;    // this obj's id number
};

int main() {
    obj a;      // output is obj 1: default constructor
    obj b(a);
    obj c = 5;
    obj d = c;
    a = c;
    b = 5;
    cout << "done!" << endl;
}
```

Please write your answer on the next page and **remove this page from the exam. This page will not be scanned for grading.**

(continued on next page)

## CSE 333 18au Midterm Exam Nov. 2, 2018

**Question 5 (cont.)** On this page, write the output produced when the program from the previous page is executed. It does compile and execute successfully.

Note that when an object is constructed, the constructor stores a unique integer `id_` number, and operations on each object print out that object's `id_` number when they are executed. The first object's `id_` number is 1, and each new object has an `id_` number that is 1 greater than the previous object.

Also note that the constructors and assignment operations ignore their arguments. That, of course, would not happen in real code, but for this question it was done to save space since the values of the arguments are not needed to trace the program's execution.

The first output line is written for you. Write the rest of the program's output after that.

Output:

```
obj 1: default constructor
```

## CSE 333 Final Exam 3/19/14

**Question 2.** (19 points) The following header file defines a class that holds a pair of integers and includes a constructor and functions for accessing the values.

```
#ifndef _Pair_h_
#define _Pair_h_

class Pair {
public:
    // Construct a Pair with given first and second values
    Pair(int first, int second)
        : first_(first), second_(second) { }

    // accessors: return first and second items from Pair
    int first() const { return first_; }
    int second() const { return second_; }

private:
    // instance variables
    int first_;
    int second_;
};

#endif // _Pair_h_
```

(a) (6 points) We would like to generalize this class so it can be used to store any pairs of values as long as the values have the same type (i.e., pairs of ints or pairs of strings, etc.)

Show the changes needed to make this a generic class where the element type is a type parameter instead of `int`. You should write your changes and additions in the above code. Hint: you'll need to start by adding `template <class T>` (or `template <typename T>`, which is equivalent) at the beginning of the class definition.

(continued on the next page)

## CSE 333 Final Exam 3/19/14

**Question 2. (cont.)** We would now like to add an addition (+) operator to the generic `Pair` class on the previous page. If `(a,b)` and `(c,d)` are `Pair` values, then `(a,b)+(c,d)` should yield a new `Pair` containing `(a+c, b+d)`. Neither of the original `Pair` objects should be modified. You do not need to check that addition (+) is defined on the items stored in a `Pair` – that is handled for you by the compiler when the addition operator is used.

(b) (5 points) Write the function declaration (not the implementation) to be added to the header file `Pair.h` for the new `operator+`.

(c) (8 points) Give the code to implement this new addition operator as it would appear in a separate file `Pair.cc` containing definitions of functions not implemented in `Pair.h`. Hint: the implementation needs to begin with `template <class T>`.