1. Determine the output of the following Scheme code. At each of the locations marked (a), (b), (c), (d) and (e), write the output of the corresponding line of Scheme.

(define x (list 2 4 6 8))

(define y (list 10 20 30))

(define z (list x y x (cdr x)))

(define wombat

(lambda (m n)

(let ((x (list 9 9))

(y (list 10 10))

(p (list m n)))

(+ (car x) (caar z) (cadr p)))))

(wombat 5 6) ;; (a)

((lambda (a) (wombat a a)) 6) ;; (b)

(define (wallaby a b c)

(+ 1 c (\* a b)))

(wallaby 2 3 4) ;; (c)

(define (kangaroo q)

(wallaby q q q))

(kangaroo 5) ;; (d)

(define (koala s t)

(lambda (u v)

(+ (\* s u) (\* t v))))

(define (platypus j k)

((koala j 2) k 3))

(platypus 7 4) ;; (e)

2. Consider the following Prolog code.

blue(3). blue(10). blue(12). blue(18). blue(39).

red(3). red(16). red(24). red(39).

yellow(4). yellow(12). yellow(16). yellow(24).

green(12). green(16). green(18). green(39). green(45).

strange(X) :- red(X), yellow(X).

bright(X) :- red(X), blue(X).

scary(X) :- blue(X), green(X).

warm(X) :- yellow(X), green(X).

meaningless(W,X,Y,Z) :- strange(W), bright(X), scary(Y), warm(Z).

fuzzy(W,X) :- meaningless(W,X,Y,\_), warm(Y).

spiky(W) :- meaningless(W,\_,\_,W).

dangerous(X,Y) :- fuzzy(X,Y), spiky(X).

Give the output of the following queries, assuming the above code has been consulted first. For

queries that can return multiple results, you must list all possible results.

(a) red(11).

(b) bright(X).

(c) scary(39).

(d) meaningless(16,39,16,4).

3. Consider the following Prolog code.

delicious(cake).

delicious(apple).

tasty(banana).

tasty(eggplant).

bland(tofu).

bland(potato).

interesting([H|\_]) :- delicious(H).

interesting([H|\_]) :- tasty(H).

interesting([\_|T]) :- interesting(T).

dull([]).

dull([H|T]) :- bland(H), dull(T).

complements(X,Y) :- delicious(X), bland(Y).

complements(X,Y) :- delicious(Y), bland(X).

suitable([X]) :- delicious(X).

suitable([X]) :- bland(X).

suitable([H1,H2|T]) :- complements(H1,H2),suitable([H2|T]).

Give the output of the following queries, assuming the above code has been consulted first. For queries that can return multiple results, you must list all possible results.

(a) tasty(X).

(b) interesting([tofu, banana, potato]).

(c) dull([tofu, banana, potato]).

(d) complements(apple, X).

(e) suitable([X, apple]).

(f) suitable([X, Y, apple]).

(g) suitable([X, banana, Y]).

4. Assuming the main function of the program below ends with the following statements, fill in all the missing outputs. You may use the fact that the ASCII value of `A' is 65, `B' is 66, and so on. You should assume the code is compiled and executed on a machine for which double and pointer values occupy 64 bits. If an output cannot be determined from the information given, write “unknown" as the output.

printf("%d\n", sizeof(a)); // output is \_\_\_\_\_\_\_\_\_

printf("%d\n", sizeof(b)); // output is \_\_\_\_\_\_\_\_\_

printf("%d\n", sizeof(c)); // output is \_\_\_\_\_\_\_\_\_

printf("%d\n", sizeof(u)); // output is \_\_\_\_\_\_\_\_\_

printf("%d\n", sizeof(s)); // output is \_\_\_\_\_\_\_\_\_

printf("%d\n", sizeof(p)); // output is \_\_\_\_\_\_\_\_\_

printf("%c\n", a[3]); // output is \_\_\_\_\_\_\_\_\_

printf("%p\n", b[3]); // output is \_\_\_\_\_\_\_\_\_

printf("%c\n", \*b[3]); // output is \_\_\_\_\_\_\_\_\_

printf("%c\n", \*c[3]); // output is \_\_\_\_\_\_\_\_\_

printf("%d\n", &(a[2]) - b[2]); // output is \_\_\_\_\_\_\_\_\_

printf("%g\n", u.d[2]); // output is \_\_\_\_\_\_\_\_\_

printf("%c\n", u.c[2]); // output is \_\_\_\_\_\_\_\_\_

printf("%c\n", s.c1); // output is \_\_\_\_\_\_\_\_\_

printf("%c\n", s.c2); // output is \_\_\_\_\_\_\_\_\_

printf("%c\n", p->c1); // output is \_\_\_\_\_\_\_\_\_

printf("%c\n", p->c2); // output is \_\_\_\_\_\_\_\_\_

printf("%c\n", q->c2); // output is \_\_\_\_\_\_\_\_\_

printf("%c\n", r->c2); // output is \_\_\_\_\_\_\_\_\_

#include "stdio.h"

#define ARR\_SIZE 4

typedef char\* charPtr;

typedef charPtr charPtrArr[ARR\_SIZE];

typedef char charArr[ARR\_SIZE];

typedef double doubleArr[ARR\_SIZE];

typedef union {

charArr c;

doubleArr d;

} charDoubleUnion;

typedef struct {

char c0;

char c1;

char c2;

char c3;

} charStruct;

int main() {

charArr a;

charPtrArr b;

char\* c[ARR\_SIZE];

charDoubleUnion u;

charStruct s;

charStruct\* p;

charStruct\* q;

charStruct\* r;

int i;

for(i=0; i<ARR\_SIZE; i++) {

a[i] = 65+i;

b[i] = a;

c[i] = a+i;

u.d[i] = ((double) i) / 10.0;

}

for(i=0; i<ARR\_SIZE; i++) {

u.c[i] = a[i];

}

s.c0 = 'W'; s.c1 = 'X'; s.c2 = 'Y'; s.c3 = 'Z';

p = &s;

p->c2 = 'M';

q = (charStruct\*) &a;

r = (charStruct\*) &u;

// Remainder of program is printfs. Omitted here.

// ...

}

5. What is the output of the following C++ program?

#include <iostream>

using namespace std;

class Banana {

public:

int a, \*b;

Banana(int aVal, int &bVal) {

a = aVal;

b = new int;

\*b = bVal;

aVal++;

bVal++;

}

virtual int sum() {return a + \*b;};

virtual ~Banana() {delete b;};

};

class Lemon : public Banana {

public:

int \*c;

Lemon(int aVal, int &bVal, int \*cVal) : Banana(aVal, bVal) {

c = cVal;

(\*cVal) += 100;

}

virtual int sum() {return a + \*b + \*c;};

virtual ~Lemon() {};

};

int main(int argc, char\*\* argv) {

int a = 10;

int b = 20;

int c = 50;

Banana\* banana1 = new Banana(a, b);

Banana\* lemon1 = new Lemon(a, b, &c);

Lemon\* lemon2 = new Lemon(a, b, &c);

Banana\* banana2 = new Banana(a, b);

Banana\* banana3 = new Banana(a, b);;

Banana\* banana4 = new Banana(a,b);;

cout << banana1->sum() << endl;

cout << banana2->sum() << endl;

cout << lemon1->sum() << endl;

cout << lemon2->sum() << endl;

cout << banana3->sum() << endl;

cout << banana4->sum() << endl;

delete banana1;

delete banana2;

delete lemon1;

delete lemon2;

}

6. What is the output of the following C++ program?

#include<iostream>

class foo {

public:

foo() {}

virtual void print1() {std::cout << "class foo" << std::endl;}

void print2() {std::cout << "class foo" << std::endl;}

};

class bar: public foo {

public:

bar() {}

virtual void print1() {std::cout << "class bar" << std::endl;}

void print2() {std::cout << "class bar" << std::endl;}

};

int main() {

foo \*fooptr;

foo f;

bar b;

fooptr = new foo();

fooptr->print1();

fooptr= new bar();

fooptr->print1();

fooptr->print2();

f.print1();

f.print2();

b.print1();

}