Consider the C program shown below. For each of the six pointer assignment statements (PASs) annotated with comments ("stmt 1", "stmt 2", ..., "stmt 6"), clearly indicate which are legal PASs, and which are not legal PASs.

```c
#include <stdio.h>

int main(void) {
    char *p1 = NULL, *p5 = NULL;
    int *p2 = NULL, *p3 = NULL, *p4 = NULL, *p6 = NULL;
    int a[5] = {1, 2, 3, 4, 5};
    char c = '5';

    p1 = &('7');        /* stmt 1 */
    p2 = &(a[3]);       /* stmt 2 */
    p3 = &10;           /* stmt 3 */
    p4 = &(a[3 - 2]);   /* stmt 4 */
    p5 = &c;            /* stmt 5 */
    p6 = &(a[2] + 1);   /* stmt 6 */

    return 0;
}
```

Indicate your solution on this page by circling Yes or No next to each of the following six questions:

stmt 1: Legal? (Yes or No)
stmt 2: Legal? (Yes or No)
stmt 3: Legal? (Yes or No)
stmt 4: Legal? (Yes or No)
stmt 5: Legal? (Yes or No)
stmt 6: Legal? (Yes or No)

Write your solution to Question (i) on this page, as described above. Please clearly indicate your solution and show all work on this page.
(ii) (5 pts.)

Consider the following C program, which includes two function definitions, including the main function.

```c
#include <stdio.h>

void sfunction(char *s, int x, int y) {
    char *p = NULL;
    p = &(s[y]);
    printf("A: %s\n", p);
    printf("B: %c\n", p[x]);
    p--;
    p--;
    printf("C: %c\n", *p);
}

int main(void) {
    char *s1 = "Maryland";
    char *s2 = "Terrapins";

    sfunction(s1, 2, 3);
    sfunction(s2, -1, 5);

    return 0;
}
```

Show the complete output as it appears on standard output.
Write your solution to Question (ii) on this page. Please clearly indicate your solution and show all work on this page.
(iii) (5 pts.)

Consider the following C program.

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

void display_result(int result) {
    static int index = 1;

    printf("result #%d: %d\n", index, result);
    index++;
}

int main(void) {
    char *s = "three/two/one", *p = s;
    int x = 100, iterations = 0, i = 0;

    iterations = 0;
    for (i = 0; i <= 10; i++) {
        iterations++;
    }
    display_result(iterations);

    iterations = 0;
    while (x > 0) {
        x = x / 7;
        iterations++;
    }
    display_result(iterations);

    iterations = 0;
    for (i = 10; i > 3; i--) {
        i -= 2;
        iterations++;
    }
    display_result(iterations);

    iterations = 0;
    while (isalpha(*p)) {
        p++;
        iterations++;
    }
    display_result(iterations);

    iterations = 0;
    for (i = 10; (i % 5) != 0; i++) {
        iterations++;
    }
    display_result(iterations);

    return 0;
}
```

Show the complete output as it appears on standard output.
Write your solution to Question (iii) on this page. Please clearly indicate your solution and show all work on this page.
(iv) (5 pts.)

Consider the following structure definition for constructing linked lists of bank records, where each record holds the information associated with a single bank account.

```c
struct bank_record {
    char *first_name;
    char *last_name;
    int id;

    /* The dollar balance of the account in cents */
    int balance;

    /* Pointer to the next list element */
    struct bank_record *next;
};
```

Consider also the following function prototype and associated header comment.

```c
/************************************************
*********************/
Given a list of bank records, pointed to by the "head" argument, determine if there is a record in the list whose id matches the "id" argument. If there are multiple records whose ids match the "id" argument, then display a meaningful error message to standard error and return a null pointer. If there is exactly one record whose id matches the "id" argument, then remove that matching record from the list and return a pointer to the matching record. If there is no record whose id matches the "id" argument, then return a null pointer (without displaying any message). Assume that the first element in the list (the element pointed to by the "head" argument) is a "dummy" record whose purpose is only to mark the beginning of the list, and assume that the "id" argument does not match the id of this dummy record.

/*********************/
struct bank_record *delete_record(struct bank_record *head, int id);

/*********************/
```

Note that no error checking is required in this function beyond what is stated in the header comment. For example, it can be assumed that the given "head" pointer is non-NULL.

Develop a complete C code implementation of the function `delete_record`.
Write your solution to Question (iv) on this page. Please clearly indicate your solution and show all work on this page.
C Reference Card (ANSI)

Program Structure/Functions

type fn(c(type1, ...)); function prototype
int main(void) { main routine
declarations
} statements

C Preprocessor
include library file
include user file
replacement text
Example. define max(A, B) ((A)>(B) ? (A) : (B))
define name(var) text
Example. define msg(A) printf("%s = %d", #A, (A))

Data Types/Declarations

class (1 byte)
integer
real number (single, double precision)
short (16 bit integer)
long (32 bit integer)
double number (64 bit integer)

Data Types/Declarations

class (1 byte)
integer
real number (single, double precision)
short (16 bit integer)
long (32 bit integer)
double number (64 bit integer)

Determine if the type is a string; if yes, call subroutine to check

Flow of Control
statement terminator
labels
in

Flow Constructions
if statement
else if (expr) statement
else statement
while (expr)
statement
for statement
statement
do statement
statement
while(expr);
switch statement
switch (expr) {

ANSI Standard Libraries
<assert.h> <ctype.h> <errno.h> <float.h> <limits.h>
<locale.h> <math.h> <setjmp.h> <signal.h> <stdarg.h>
<stdio.h> <stdlib.h> <string.h> <time.h>

Character Class Tests <ctype.h>

String Operations <string.h>

s is a string; ca, ct are constant strings

Operators (grouped by precedence)

++
-+-
*++++
/+-+
%--

conditional expression

expression evaluation separator

Unary operators, conditional expression and assignment operators
from group right to left; all others group left to right.

c 2007 Joseph H. Silverman Permissions on back. v2.2
Standard Utility Functions  <stdlib.h>

absolute value of int n    abs(n)
absolute value of long n    labs(n)
quotient and remainder of ints n,d    div(n,d)
returns structure with div_t.quot and div_t.rem
quotient and remainder of longs n,d    ldiv(n,d)
returns structure with ldiv_t.quot and ldiv_t.rem
pseudo-random integer [0,...,RAND_MAX]    rand()
set random seed to n    srand(n)
terminate program execution    exit(status)
pass string s to system for execution    system(s)
Conversions
convert string s to double    atof(s)
convert string s to integer    atoi(s)
convert string s to long    atol(s)
convert prefix of s to double    strtod(s,&endp)
convert prefix of s (base b) to long    strto1d(s,&endp,b)
same, but unsigned long    strtoul(s,&endp,b)
Storage Allocation
allocate storage    malloc(size), calloc(nobj,size)
deallocate storage    free(ptr);
Array Functions
search array for key    bsearch(key,array,n,size,cmpf)
sort array ascending order    qsort(array,n,size,cmpf)

Time and Date Functions <time.h>

processor time used by program    clock()
Example. clock()/CLOCKS_PER_SEC is time in seconds
current calendar time    time()
structure for calendar time types    tm

short time in seconds (double)    difftime(time2,time1)

arithmetical times representing times    clock_t,time_t
structure for calendar time types    tm

seconds after minute    tm_sec
minutes after hour    tm_min
hours since midnight    tm_hour
day of month    tm_mday
months since January    tm_mon
years since 1900    tm_year
days since Sunday    tm_wday
days since January 1    tm_wday
Daylight Savings Time flag    tm_isdst
convert local time to calendar time    mktime(tp)
convert time in tp to string    strftime(tp,format,ty)
convert calendar time in tp to local time    strptime(tp,strctime(tp,format,ty)
convert calendar time to GMT    strftime(tp,format,ty)
convert calendar time to local time    localtime(tp)

format date and time info    strftime(s,format,ty)

structure for calendar time types    tm

Mathematical Functions <math.h>

Arguments and returned values are double

trig functions    sin(x), cos(x), tan(x)
inverse trig functions    asin(x), acos(x), atan(x)

arctan(y/x)    atan2(y,x)

hyperbolic trig functions    sinh(x), cosh(x), tanh(x)

exponentials & logs    exp(x), log(x), log10(x)

exponentials & logs (2 power)    ldexp(x,n), frexp(x,fp)

division & remainder    modf(x,ip), fmod(x,y)
powers    pow(x,y), sqrt(x)

ramping    ceil(x), floor(x), fabs(x)

Integer Type Limits  <limits.h>

The numbers given in parentheses are typical values for the constants on a 32-bit Unix system, followed by minimum required values (if significantly different).

CHAR_BIT bits in character

CHAR_MAX max value of signed char

CHAR_MIN min value of signed char

CHAR_MAX or 0)

SCHAR_MAX or 0)

SCHAR_MIN min value of signed char

SHRT_MAX max value of short

SHRT_MIN min value of short

INT_MAX max value of int

INT_MIN min value of int

LONG_MAX max value of long

LONG_MIN min value of long

ULONG_MAX max value of long


FILE * declare file pointer

FILE * fopen("name","mode")

read file from 

fscanf(fp,format,arg1,...)

write to file

fprintf(fp,"format",arg1,...)

read and store n elts to *ptr

freed(*ptr,eltsize,n,fp)

write n elts from *ptr to file

fwrite(*ptr,eltsize,n,fp)

close file

fclose(fp)

non-zero if error

ferror(fp)

non-zero if already reached EOF

feof(fp)

read line to string s (< max chars)

fgets(s,max,fp)

write string s

fputs(s,fp)

Codes for Formatted I/O: "%+ = 0:pmn"

* left justify
+ print with sign
space print space if no sign
0 pad with leading zeros
w min field width
p precision
m conversion character:
h short, l long, L long double
c conversion character:
d,i integer u unsigned
c single char s char string
e float (printf) e,E exponential
f float (scanf) f,l float (scanf)
o octal x,X hexadecimal
p pointer n number of chars written
g,G same as f or e,E depending on exponent

Variable Argument Lists  <stdarg.h>

declaration of pointer to arguments

va_list ap;
initialization of argument pointer

va_start(ap,lastarg);
lastarg is last named parameter of the function
access next unnamed arg, update pointer

va_arg(ap,type)
call before exiting function

va_end(ap);

January 2007 v2.2. Copyright © 2007 Joseph H. Silverman
Permission is granted to make and distribute copies of this card provided the copyright notice and this permission notice are preserved on all copies.
Send comments and corrections to J.H. Silverman, Math. Dept., Brown Univ., Providence, RI 02912 USA. (jhs@math.brown.edu)
Problem 1:

stmt 1, stmt 3, and stmt 6 are NOT legal.

Problem 2:

A: yland
B: a
C: a
A: pins
B: a
C: r

Problem 3:

result #1: 11
result #2: 3
result #3: 3
result #4: 5
result #5: 0
Problem 4:

```c
struct bank_record *delete_record(struct bank_record *head, 
    int id) {
    struct bank_record *p = NULL; 
    struct bank_record *last_match; 
    int matches = 0;

    for (p = head; p != NULL; p = p->next) {
        if (p->id == id) {
            matches++;
            last_match = p;
        }
    }

    if (matches == 0) {
        return NULL;
    } else if (matches > 1) {
        fprintf(stderr, "Error: multiple accounts have ");
        fprintf(stderr, "the same ID\n");
        return NULL;
    } else {
        for (p = head; p->next != last_match; p = p->next);
        p->next = last_match->next;
        last_match->next = NULL;
        return last_match;
    }
}
```