CSC 111 Fall 2012 Solutions — Dr. H.A. Müller
Final Examination
University of Victoria

Your Name Hausi Müller **UVicID**

Instructions

- This final examination consists of 13 pages and 28 questions.
- The questions are worth 2, 4, 6 or 8 points, for a total of 100 points. The points are listed in square brackets at the end of the first line of each question.
- You have 3 hours for this examination. **Time management: approx. 4-7 minutes per question.**
- This final exam is closed-books, closed-notes, no calculators, no gadgets, and no electronic devices.
- Turn in your completed final exam at the front of the class.
- For multiple choice questions, mark all circles that are next to correct choices.
- **Do not leave before 10:30 am.**

1. What do the names FORTRAN, LISP, ALGOL, and COBOL represent? **[2]**

   ✓ Programming languages invented and defined in the Fifties

   ◯ Programming languages invented and defined in the Sixties

   ◯ Programming languages invented and defined in the Seventies

   ◯ Programming languages invented and defined in the Eighties

2. Which one is not a characteristic language feature of the C programming language? **[2]**

   ◯ Rich operator set

   ◯ Famous I/O library

   ◯ Ideal for systems programming

   ✓ Automatic garbage collection

3. Which of the following printf() format specifications prints a pointer in hexadecimal format in the programming language C? **[2]**

   ◯ %-4d

   ◯ %.2f

   ✓ %p

   ◯ %s

4. Consider a pointer p that can point to a list node. How is the following English statement translated into the programming language C? **[2]**
If pointer p is NOT equal to NULL AND its next pointer is not equal to NULL then p becomes the next pointer

- ◯ if ((p =! 0) || (p->next != 0)) { p = p-> next;}
- ◯ if ((p =! NULL) & (p->prev != NULL)) { p = p-> prev;}
- ◯ if ((p =! NULL) | (p->next != NULL)) { p = p-> next;}
- ✓ if ((p =! NULL) && (p->next != NULL)) { p = p-> next;}

5. What is the output of the following syntactically correct C program? **[4]**

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

void magicWand(int* c, int d) {
   printf ("%d ", *c);    printf ("%d ", d);
   *c = *c * 4 + d;
   printf ("%d ", *c);    printf ("%d ", d);
} /* magicWand */

int main(void) {
   int x = 21;    int y = 19;
   printf ("%d ", x);    printf ("%d ", y);
   magicWand (&x, y);
   printf ("%d ", x);    printf ("%d\n ", y);
   return EXIT_SUCCESS;
} /* main */
```

- ◯ 21 19 21 19 82 19 82 19
- ◯ 21 19 19 21 82 19 82 19
- ✓ 21 19 21 19 103 19 103 19
- ◯ 21 19 21 19 82 19 21 19

6. Which sequence of operators has the correct precedence order—from highest to lowest—in the programming language C? **[2]**

- ✓ () < && +=
- ◯ [] ! %= <=
- ◯ % + == ->
- ◯ ++ % || []

7. What is the console output produced by the following C program? **[2]**

```
#include <stdio.h>
#include <stdlib.h>
int main(void) {
        int x = 19;
        int a = 1;
        x  -=  3  * x  +  ( a = 18 );
        printf ("%d\n", x);
        return EXIT_SUCCESS;
} /* main */
```

◯ -57

✓ -56

◯ 56

◯ 57

8. What is the console output produced by the following C program? **[2]**

```
#include <stdio.h>
#include <stdlib.h>
int main(void) {
        int k = 9;
        while (k > 8) {
           if (k % 2 == 1) printf("%d ", k);
           k = k + 1;
           if (k > 27) break;
        }
        for (k=5;  k>6;  k=k-1) printf("%3d",  k);
        printf("\n");
        return EXIT_SUCCESS;
} /* main */
```

◯ 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27

✓ 9 11 13 15 17 19 21 23 25 27

◯ 9 11 13 15 17 19 21 23 25 27 29 31 33 35 … (infinite loop)

◯ 9 10 11 12 13 14 15 16 17 18 19 20 21 22 … (infinite loop)

9. Find the memory access error in the following C program? **[4]**

```
#include <stdio.h>
#include <stdlib.h>
int main(void) {
        int a, b, c;
        int* x;
        int* y;
        int** z;
        a = 17;
        z = &y;
        *z = x;
        b = *y;
        c = a*b;
        printf("c = %d\n", c);
        return EXIT_SUCCESS;
} /* main */
```

◯ z = &y;

✓ *z = x;

✓ b = *y;

◯ c = a*b;

10. What is the output of the following syntactically correct C program? **[4]**

```
#include <stdio.h>
#include <stdlib.h>
int main(void) {
        int z = 0;
        int n = 11;
        int k = 1;
        int* p = &k;
        while (*p <= n) {
            z = z + *p;
            *p = *p + 2;
        } /* while */
        printf("n = %d  z = %d\n", n, z);
        return EXIT_SUCCESS;
} /* main */
```

◯ n = 11 z = 72

✓ n = 11 z = 36

◯ n = 11 z = 66

◯ n = 11 z = 40

11. What is the output of the following syntactically correct C program? **[4]**

```
#include <stdio.h>
#include <stdlib.h>
#include <stdbool.h>
int main(void) {
        int a, d;
        int *b, *c;
        a = 17;
        b = &a;
        c = b;
        bool b1 = (b == c);
        bool b2 = (&a == c);
        bool b3 = (b == &d);
        bool b4 = (*b == 19);
        // printf("b==c  is ");
        if (b1) printf("true "); else printf("false ");
        // printf("&a == c  is ");
        if (b2) printf("true "); else printf("false ");
        // printf("b == &d  is ");
        if (b3) printf("true "); else printf("false ");
        // printf("*b == 19  is ");
        if (b4) printf("true "); else printf("false ");
        printf("\n");
        return EXIT_SUCCESS;
} /* main */
```

◯ true false false true

◯ false true true false

✓ true true false false

◯ true false true false

12. In the following table, complete the columns for program statement and data type according to the appropriate construction patterns in the left-most column. **[4]**

Pattern	Program Statement	Data Type
Atomic Element	Assignment	Simple
Enumeration	Compound	Struct
Repetition by known factor	for	Array
Repetition by unknown factor	while or do while	File

13. Consider the following syntactically correct C declarations and assignments. **[8]**

```
int x;
int y;
int *p;
int *q;
int** t;
x = 44;   p = &x;
q = p;    y = 19;
t = &q;
```

What are the values of the following expressions (i.e., true or false)?

(&x == p)	true
(p == &y)	false
(*q == 17)	false
(**t== *p)	true
(y == x)	false
(*q == 44)	true
(*q == *p)	true
(&x == *t)	true

14. What is the output of the following syntactically correct C program? **[4]**

```
#include <stdio.h>
#include <stdlib.h>
int main(void) {
   int k = 5;
   while (k < 12) {
        printf("%d ",  k%7);
        k = k + 1;
   } /* while */
   printf("\n");
   return EXIT_SUCCESS;
} /* main */
```

Output: `5 6 0 1 2 3 4`

15. Consider the following declarations: **[4]**

```
typedef struct {
        int year;
        int month;
        int day;
} Date;
Date dob;
Date *d = &dob;
```

Using variable **d** initialize **dob** with the birthday **July 1, 1867.**

```
d->month = 7;
d->day = 1;
d->year = 1867;
```

16. What is the output of the following syntactically correct C program? **[4]**

```
#include <stdio.h>
int main(void) {
    int k = 9;
    while (k < 10) {
        printf("%d ", k);
        k = k - 1;
    }
    printf("\n");
    return 0;
} /* main */
```

Output:

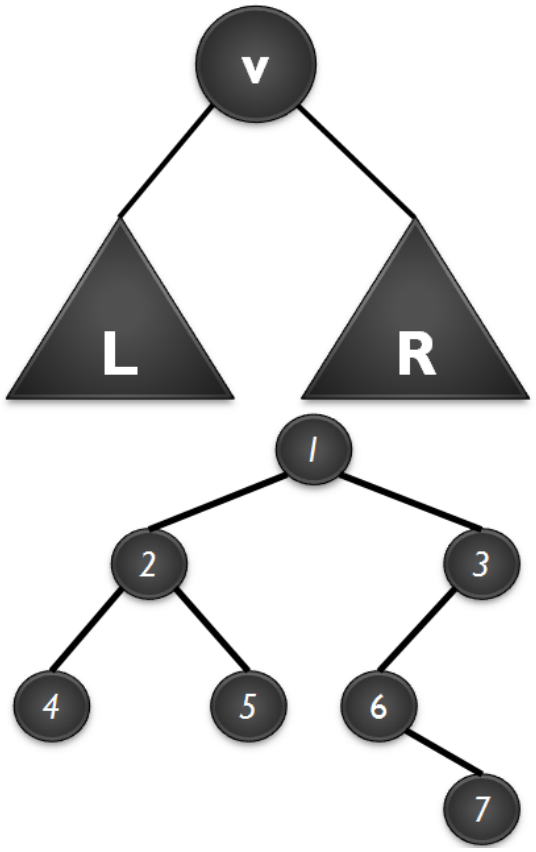
```
9 8 7 6 5 4 3 2 1 0 -1 -2 -3 -4 -5 -6 -7 -8 -9 -10  -11 -12 -13 …
```

17. What is the output of the following syntactically correct C program? **[4]**

```
#include <stdio.h>
#include <stdlib.h>
int main(void) {
    int k;
    for (k=81;  k>0;  k=k-17) if (k % 3 != 0) printf("%3d",  k);
    printf("\n");
    return EXIT_SUCCESS;
} /* main */
```

Output: 64 47 13

18. Consider the following binary tree. In what order are the Nodes 1 through 7 visited using the *preorder,inorder and postorder* binary tree traversal algorithms discussed in class? **[6]**



Preorder: v, L, R

1 2 4 5 3 6 7

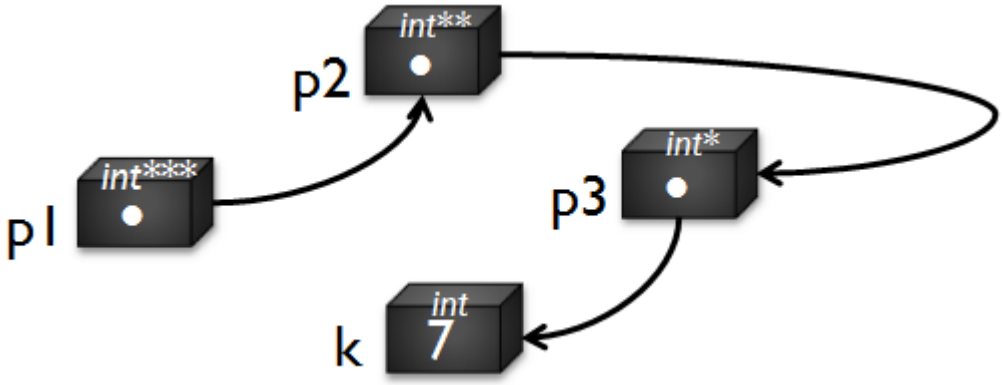
Postorder: L, R, v

4 5 2 7 6 3 1

Inorder: L, v, R

4 2 5 1 6 7 3

19. Realize the following memory configuration using C variable declarations and pointer assignments. Write three assignments using pointers p1, p2, and p3 to store 7 in variable k. **[4]**




```
int k;
int* p3;
int** p2;
int*** p1;

p3 = &k;
p2 = &p3;
p1 = &p2;

***p1 = 7;   **p2 = y;   *p3 = 7;
```

20. Which statement is incorrect? **[2]**

- ◯ The main operations of a stack are push(), pop() and top().
- ◯ The main operations of a queue are enqueue(), dequeue() and first().
- ✓ The operations of a stack and a deque are subsets of the operations of queue.
- ◯ A deque provides operations to insert and delete elements at both ends of the list.

21. What is a binary tree? **[2]**

- ◯ A special case of a tree that stores only binary values (i.e., 0's and 1's).
- ◯ A forest consisting of 2 trees.
- ◯ A data structure where nodes are linked to other nodes using a linked list.
- ✓ A special case of a tree where each node has 0, 1 or 2 children.

22. Write a syntactically correct C function to swap the values of two integer variables. **[4]**

```
void swap(int* a, int* b) {
    int temp = *a;   *a = *b;   *b = temp;
} /*swap*/
```

23. Write a syntactically correct C function to shift the elements of a one-dimensional array one position to the right. The last element of the array—the one that drops off the array—is stored in the first element of the array—the one that was freed. **[6]**

```
#include <stdio.h>
void shiftArrayRight(int a[],  int len) {
        /* ... Your code goes here */
    int first = a[len-1];
    int k;
    for (k = len-1; k >0; k--) a[k] = a[k-1];
    a[0] = first;









} /* shiftArrayRight */
```

24. What is the difference between call-by-value and call-by-reference parameter passing in the programming language C? Explain in your own words. **[6]**

- The main parameter passing mechanism in C is call-by-value
- Call-by-reference has to be simulated using pointers in C
- Arrays are passed using call-by-reference in C
- Most languages explicitly support call-by-value and call-by-reference
- Fortran: call-by-reference and by value-result
  Pascal, C++:, Java, C# call-by-reference and call-by-value

- **Def. Call-by-value.** Variables that are passed to a function using call-by-value cannot be changed by the function.
  Call-by-value parameters are said to be *input only.*
- **Def. Call-by-reference.** Variables that are passed to a function using call-by-reference can be changed by the function. Call-by-reference parameters are said to be
  input and output.

25. What data structure can be represented with the following C declarations? **[2]**

```
typedef struct  {  int info;  }  Item;
typedef struct Item*  ItemRef;
```

```
typedef struct NodeStruct*  NodeRef;
typedef struct NodeStruct  {
    ItemRef  item;
    NodeRef  left;
    NodeRef  right;
} Node;
```

◯ Graph

✓ Doubly linked list

◯ Binary tree

◯ N-ary tree

26. Write a syntactically correct C function to convert every lowercase character in the string s to uppercase (i.e., capitalizes the string) and store the modified character back into s. The standard library functions (defined in ctype.h) 'islower', 'isupper', 'tolower' and 'toupper' may be helpful to implement this function. **[4]**

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

void capitalizeString(char *s) {
        /* ... Your code goes here */
   while (*s != '\0') {
      if (islower(*s)) *s = toupper(*s);
      s++;
   } /*while*/









} /* capitalizeString */
```

27. Consider the following syntactically correct function of inserting a node into a singly linked list. Assuming that head is NULL, which of the following statements is correct? **[2]**

```
void insertNode(SlistRef sl, Info g) {
        NodeRef x = initializeNode(g);
        sl->size++;
        if (sl->head == NULL) {
                sl->head = x;    x->next = NULL;
        } else {
                x->next = sl->head;    sl->head = x;
        } /*if*/
} /* insertNode */
```

✓ head points to a newly created node

◯ next of node x points to the first node

◯ head points to null

◯ none of the above

28. Write a syntactically correct C main() function to generate 100 random numbers in the range of 32 to 63. Use the function genRand() below that uses the standard library function rand() to generate a single random number. **[4]**

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

int genRand(int base, int range) {
    return rand() % range + base;
} /*genRand*/

int main(void) {
    srand(time(NULL));  // seed random number generator
    /* your code code here */

    int k;
    int rn;
    for (k=0; k<100; k++) {
        rn = genRand(32, 32);
        printf("%d = %d\n", k, rn);
    } /*for*/

    return EXIT_SUCCESS;
}
```

END