CS 135501 – Introduction to Programming

Final Exam: 3:20 – 5:10 p.m., January 9, 2006

If you do not understand a question, please raise your hand. Keep your eyes on your own paper and do not talk during the exam.

1. (10 points) Write TRUE or FALSE for each of the following questions.

F (a) The standard output stream is normally connected to the keyboard.

F (b) The conversion specifier **`u`** is used to display unsigned integers in hexadecimal form.

T (c) The tag name of a structure is optional.

F (d) Members of different structures must have unique names.

F (e) If the file position pointer points to a location in a sequential file other than the beginning of the file, the file must be closed and reopened to read from the beginning of the file.

F (f) Function **`fseek`** can only seek relative to the beginning of a file.

T (g) A tree node that has no children is called a leaf node.

T (h) A stack is referred to as an LIFO (last-in, first-out) data structure because the last element inserted is the first one removed.

T (i) UNIX utility **`make`** reads a file called **`makefile`** that contains instructions for compiling and linking a program consisting of multiple source files.

F (j) Function **`signal`** can be used to generate a signal within a program.

2. (40 points) Pick the most appropriate choice for each question.

(a) A string array

(1) stores an actual string in each of its elements

(2) can only provide access to strings of a certain length

(3) is actually an array of pointers

(4) is always less memory efficient than an equivalent double-subscripted array

(b) Which statement about the parameter definition

```
int (*compare)( int a, int b)
```

is false?

(1) It defines a parameter that is a pointer to a function that receives two integer arguments and returns a pointer to an integer as a result.

(2) Parentheses are needed around *****compare because ***** has a lower precedence than the parentheses enclosing the function parameters.

(3) Without the parentheses it would have defined a function that receives two integers and returns a pointer to an integer.

(4) The corresponding parameter in the function prototype would ordinarily be

```
int (*)( int, int )
```

(c) The **`isxdigit`** function would return false on

(1) **'a'**

(2) **'A'**

(3) **'2'**

(4) **'g'**

(d) Which function does not read data from standard input?

(1) **scanf**

(2) **sscanf**

(3) **sprintf**

(4) **getchar**

(e) The **strlen** function returns

(1) the number of characters in a string excluding the null character

(2) the number of characters in a string including the null character

(3) the ASCII representation of the character

(4) none of these

(f) Which is not a formatting capability of **printf**?

(1) left justification

(2) centering

(3) right justification

(4) aligning a column of numbers so that decimal appoints appear one above the other

(g) What would be the output of the following statements?

```
char* value = “hello”;
printf(“%s”, value);
```

(1) h

(2) hello

(3) value

(4) none of these

(h) Which is not an input formatting capability of **scanf**?

(1) inputting all types of data

(2) inputting specific characters from an input stream

(3) skipping specific characters in the input stream

(4) replacing specific characters in the input stream

(i) The expression **aptr->suit** is equivalent to

(1) **aptr.suit**

(2) ***aptr.suit**

(3) **(*aptr).suit**

(4) ***aptr.(suit)**

(j) Evaluate **(00001000 & 11000101) ^ (11110000)**

(1) **00111101**

(2) **11000000**

(3) **00111101**

(4) **11110000**

(k) Which of the following is not a stream associated with C files?

(1) **stdin**

(2) **stdout**

(3) **stdchar**

(4) **stderr**

(l) Function **fwrite**

(1) is equivalent to function `fprintf`
(2) transfers a specified number of bytes beginning at a specified location in memory to a location in a file indicated by the file position pointer
(3) transfers a specified number of bytes beginning at a specified location in memory to a location in a file indicated by one of its arguments
(4) is equivalent to function `fprintf`, except that `fwrite` can only write to standard streams

(m) Which statement is true?
(1) `fopen` returns a `FILE` structure.
(2) `fopen` returns a pointer to a `FILE` structure.
(3) `fopen` returns a file control block (FCB).
(4) `fopen` returns a pointer to a file control block (FCB).

(n) How many pointers which are not NULL in a doubly linked list with five nodes?
(1) 5
(2) 8
(3) 15
(4) 10

(o) For a non-empty linked list, select the code that should appear in a function that adds a node to the beginning of the list. `newPtr` is a pointer to the new node to be added, and `firstPtr` is a pointer to the current first node. Each node contains a pointer `nextPtr`, a link to the next node.

(1)
```
firstPtr->nextPtr = newPtr;
firstPtr = newPtr;
```
(2)
```
firstPtr = newPtr;
firstPtr->nextPtr = newPtr;
```
(3)
```
newPtr->nextPtr = firstPtr;
firstPtr = newPtr;
```
(4)
```
firstPtr = newPtr;
newPtr->nextPtr = firstPtr;
```

(p) Which include statement is usually appropriate for standard library files?
(1) `include` <filename>
(2) `#include` "filename"
(3) Both (a) and (b) are appropriate.
(4) Neither (a) nor (b) is appropriate.

(q) Large portions of code can be prevented from compiling by
(1)
```
#if 0
code prevented from compiling
  #endif
```
(2)
```
 #nodefine
     code prevented from compiling
  #endif
```
(3)
```
#if 1
     code prevented from compiling
  #endif
```
(4) `#ifndef 0`

code prevented from compiling

```
#endif
```

(r) Which of the following symbols is used in UNIX for redirecting input?

(1) `<`

(2) `|`

(3) `$`

(4) `>>`

(s) Which of the following function prototypes is correct?

(1) `double average( int, ... )`

(2) `double average( ..., int );`

(3) `double average( int, ... );`

(4) `double average( int, ... , int);`

(t) Preprocessing occurs

(1) before a program is compiled.

(2) during compilation.

(3) after compilation but before execution.

(4) immediately before execution.

3. (10 points)

(a) A stack is initially empty, and then the following commands (in pseudo-code) are performed.

```
push 1
pop
push 3
push 5
pop
```

What is the number at the top of the stack? 3

(b) A queue receives the following commands (in pseudo-code):

```
enqueue 1, 3, 8, 4
dequeue 2 elements
enqueue 5, 6
dequeue 1 elements
```

What is the number at the tail of the queue? 6

(c) Add the following nodes to a binary search tree in the order they appear.

1 2 3 4 5

What is the the postorder traversal of this tree? Separate each number with a blank in your answer. 5 4 3 2 1

(d) Find the error in each of the following program segments and explain how to correct it.

(i) `printf(“%c\n,” “This is a test”);` `%c -> %s`

(ii) `printf(“%d\n”, 1.234);` `%d -> %f`

4. (10 points)

(a) Write a statement or preprocessor directive to accomplish each of the following tasks.

(i) Open the file **test.dat** for writing and assign the returned file pointer to **testPtr**. testPtr = fopen(“test.dat”, ”w”);

(ii) Define macro **MIN2** which receives two arguments **x** and **y**, and determines the smallest between **x** and **y**. #define MIN2(x, y) (x < y ? x : y)

(iii) Define macro **MIN3** which receives three arguments **u**, **v**, and **w**, and determines the smallest among **u**, **v**, and **w**. Macro **MIN3** should use macro **MIN2** defined in part (ii) to determine the smallest value.

#define MIN3(u,v,w) (MIN2 (u , MIN2 (v, w)))

(b) Given the following structure definitions and variable declarations,

```
struct customer {
  char name [ 15 ];
  int customerNumber;
  struct {
    char phoneNumber [ 10 ];
    char zipCode [ 5 ];
  } personal;
} customerRecord, *customerPtr;
customerPtr = &customerRecord;
```

write a separate expression that can be used to access the structure member in each of the following parts.

(i) Member **phoneNumber** of member **personal** of structure **customerRecord**.

customerRecord.personal.phoneRumber

(ii) Member **zipCode** of member **personal** of the structure pointed to by **customerPtr**.

customerPtr -> personal.zipCode

5. (10 points) Write a program that inputs a line of text, tokenizes the line with function **strtok** and outputs the tokens in reverse order. Assume tokens are separated with space characters. (The detailed definition of **strtok** is in appendix.)

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

void reverseTokens( char *sentence );

int main()
{
   char text[ 80 ];

   printf( “Enter a line of text:\n” );

   gets( text );

   reverseTokens( text );
}
```

```
void reverseToekns( char *sentence )
{
    char *pointers[ 50 ];
    char *temp;
    int count = 0;
    int i;

    temp = strtok( sentence, “ “ );
    while( temp )
    {
          pointers[ count++ ] = temp;
          temp = strtok( NULL, “ “ );
    }

    printf( “The tokens in reverse order are:\n” );

    for( i = count – 1; i >= 0; i++ )
    {
          printf( “%s “ , pointers[ i ] )’
    }
}
```

6. (10 points) Suppose that your computer uses 16-bit unsigned integers. Write a program that inputs two characters (each of which is stored by 8 bits in your computer) from the keyboard, and concatenates the two characters into an unsigned integer variable. After the concatenation, the leftmost 8 bits of the unsigned integer are from the first character, and the rest of the bits are from the second character. Your program should output the unsigned integer (after concatenation) in the 16-bit format. Both the left-shift and bitwise inclusive OR operators must be used in your program.

```
#include <stdio.h>

unsigned packCharacters( char x, char y );
void displayBits( unsigned value );

int main()
{
    char a;
    char b;
    unsigned result;

    printf( “Enter two characters: “ );
    scanf( “%c %c” , &a, &b );

    result = packCharacters( a, b );

    displayBits( result );
    return 0;
}

unsigned packCharacters( char x, char y )
{
    unsigned pack = x;

    pack <<= 8;
    pack |= y;
    return pack;
}

void displayBits( unsigned value )
{
    unsigned c;
    unsigned displayMask = 1 << 15;

    printf( “%7u = “, value );

    for( c = 1; c <= 16; c++ )
    {
          value & displayMask ? putchar( ‘1’ ) : putchar( ‘0’ );
          value << = 1;

          if( c % 8 )
          {
                    putchar( ‘ ‘ );
          }
    }

    putchar( ‘\n’ );
}
```

7. (10 points) Given a binary tree, let its root be at level 1. If a node is at level *x*, then each child of the node is at level *x*+1. The height of a binary tree is defined to be the maximum level of any node in the tree. Write a function that receives a pointer to the root of a binary tree and determines the height of the tree. Assume each node in a binary tree has the following structure:

```
struct treeNode {
  struct treeNode *leftPtr; /* pointer to left subtree */
  int data; /* node value */
  struct treeNode *rightPtr; /* pointer to right subtree */
};
```

```
int height(struct treeNode *ptr)
{
      int temp[2];
```

```
    if (ptr == NULL)
        return 0;
    else
        return 1 + ((temp[0] = height(ptr->leftPtr)) > (temp[1] = height(ptr->rightPtr))) ?
temp[0] : temp[1];
}
```

Appendix：

```
char *strtok( char *s1, const char *s2 );
```

A sequence of calls to **`strtok`** breaks string `s1` into “tokens”—logical pieces such as words in a line of text—separated by characters contained in string `s2`. The first call contains `s1` as the first argument, and subsequent calls to continue tokenizing the same string contain `NULL` as the first argument. A pointer to the current token is returned by each call. If there are no more tokens when the function is called, `NULL` is returned.

1, 4, 7 題請找李宗賢助教
2, 5 題請找林詠嘉助教
3, 6 題請找吳珮琦助教