

Princeton University

COS 217: Introduction to Programming Systems

Fall 2007 Midterm Exam Answers

The exam was a 50-minute, open-book, open-notes exam.

Question 1 Part 1

```
((0xFA / 02) >> 1) & 017) - 07
```

Convert to 8-bit binary:

```
= (((11111010 / 00000010) >> 00000001) & 00001111) - 00000111
= ((01111101 >> 00000001) & 00001111) - 00000111
= (00111110 & 00001111) - 00000111
= 00001110 - 00000111
= 00000111
= 7 (decimal)
```

Question 1 Part 2

5

Question 1 Part 3

-1

Question 1 Part 4

The data are abstract. Clients cannot "see" the data. Clients can access the data only by calling the functions that encapsulate it.

Question 1 Part 5

If the stack and the heap grew in the same direction, then it would be possible for one to be exhausted while the other still has memory available. The stack and heap grow toward each other to eliminate that possibility.

Question 1 Part 6

Cast to a pointer to a function that accepts a pointer to an object of unspecified type, an integer, and a pointer to a character, and returns nothing.

Question 1 Part 7

```
int hash(const char *key)
{
    return 0;
}
```

A good hash function places bindings uniformly over buckets. That hash function is particularly bad because it places all bindings in the same bucket.

Question 2

This is the DFA expressed in textual form:

```
LETTER_STATE (NOT FLOAT) (the start state)
  LETTER:  LETTER_STATE
  NUMERAL: LETTER_STATE
  DOT:     LETTER_STATE
  SPACE:   SPACE_STATE

SPACE_STATE (NOT FLOAT)
  LETTER:  LETTER_STATE
  NUMERAL: NUMERAL_BEFORE_DOT_STATE
  DOT:     DOT_STATE
  SPACE:   SPACE_STATE

NUMERAL_BEFORE_DOT_STATE (NOT FLOAT)
  LETTER:  LETTER_STATE
  NUMERAL: NUMERAL_BEFORE_DOT_STATE
  DOT:     FLOAT_STATE
  SPACE:   SPACE_STATE

DOT_STATE (NOT FLOAT)
  LETTER:  LETTER_STATE
  NUMERAL: FLOAT_STATE
  DOT:     LETTER_STATE
  SPACE:   SPACE_STATE

FLOAT_STATE (FLOAT)
  LETTER:  LETTER_STATE
  NUMERAL: FLOAT_STATE
  DOT:     LETTER_STATE
  SPACE:   SPACE_STATE
```

Question 3 Part 1

```
/*
Name: palindrome
Description: Prompts the user to enter a
string upto 15 characters long and
checks to see if it is a palindrome.
*/

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

enum {FALSE, TRUE};
enum {BUFFER_LENGTH = 15};

int checkPalindrome(char *pcForward, char *pcReverse) {
    int ispalindrome = TRUE;

    /* Check if the first and last characters are equal,
     * then move inward checking each pair of characters
     */
    while(pcForward < pcReverse && ispalindrome)
    {
        /* If corresponding characters do not match
         * then the word is not a palindrome
         */
        if(*pcForward != *pcReverse) {
            ispalindrome = FALSE;
        } /* end if */
        pcForward++;
        pcReverse--;
    } /* end while */
    return ispalindrome;
} /*end checkPalindrome*/
```

```

int main (void) {
    /* Create a character array */
    char string[BUFFER_LENGTH];
    int ispalindrome; /* boolean */
    char *pcForward,*pcReverse; /*pointers to traverse the
                                string from the front and
                                the back in the forward and
                                reverse directions respectively*/

    int c;
    int i = 0;

    /* prompt the user to input the string */
    printf("Enter a string (maximum %d characters): ", BUFFER_LENGTH - 1);
    c = getchar();
    while ((c != EOF) && (c != '\n'))
    {
        string[i] = (char)c;
        i++;
        if (i > BUFFER_LENGTH - 1)
            break;
        c = getchar();
    }
    string[i] = '\0';

    pcReverse = string + strlen(string) - 1;
    pcForward = string;

    ispalindrome = checkPalindrome(pcForward, pcReverse);

    /* print out the results */
    switch (ispalindrome) {
        case TRUE:
            printf("%s is a palindrome!\n", string);
            break;
        case FALSE:
            printf("Sorry, %s is not a palindrome\n", string);
    }
    return 0;
} /* end main */

```

Question 3 Part 2

```

/*****
Name: palindrome
Description: Prompts the user to enter a
            string upto 15 characters long and
            checks to see if it is a palindrome.
*****/

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

enum {FALSE, TRUE};
enum {INITIAL_BUFFER_LENGTH = 2};

int checkPalindrome(char *pcForward, char *pcReverse) {
    int ispalindrome = TRUE;

    /* Check if the first and last characters are equal,
     * then move inward checking each pair of characters
     */
    while(pcForward < pcReverse && ispalindrome)
    {
        /* If corresponding characters do not match
         * then the word is not a palindrome
         */
    }
}

```

```

        if(*pcForward != *pcReverse) {
            ispalindrome = FALSE;
        } /* end if */
        pcForward++;
        pcReverse--;
    } /* end while */
    return ispalindrome;
} /*end checkPalindrome*/

int main (void) {
    /* Create a character array */
    char *string;
    int ispalindrome; /* boolean */
    char *pcForward,*pcReverse; /*pointers to traverse the
                                string from the front and
                                the back in the forward and
                                reverse directions respectively*/

    int c;
    int i = 0;
    int bufferLength = INITIAL_BUFFER_LENGTH;

    string = (char*)malloc(bufferLength);

    /* prompt the user to input the string */
    printf("Enter a string: ");
    c = getchar();
    while ((c != EOF) && (c != '\n'))
    {
        if (i == bufferLength - 1)
        {
            bufferLength *= 2;
            string = (char*)realloc(string, bufferLength);
        }
        string[i] = (char)c;
        i++;
        c = getchar();
    }
    string[i] = '\0';

    pcReverse = string + strlen(string) - 1;
    pcForward = string;

    ispalindrome = checkPalindrome(pcForward, pcReverse);

    /* print out the results */
    switch (ispalindrome) {
        case TRUE:
            printf("%s is a palindrome!\n", string);
            break;
        case FALSE:
            printf("Sorry, %s is not a palindrome\n", string);
    }

    free(string);
    return 0;
} /* end main */

```

Copyright © 2007 by Robert M. Dondero, Jr.