CSE 220 – Programming in C

Quiz #2

Spring 2016

Name: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

1. Specify if each of the following statements is True or False (4 pts each).

|  |  |
| --- | --- |
| If p and q are two pointers to elements of the same array, subtracting p from q is a valid arithmetic operation | T |
| When a function is called, its parameters are local to the function. | T |
| When an array is passed to a function, copies of the array elements are made for the function to manipulate. | F |
| If p is a pointer to an int, then the expression &p is invalid and results in a compiler error. | F |
| A function can return more than one value at a time | F |

1. Consider the following array. Write statements to do the following: (15 pts)

double a[10] = {0.0, 10.0, 20.0, 30.0, 40.0, 21.0}, \*p, \*q;

|  |  |
| --- | --- |
| Set p to point to the last element of the array | p = a+9; |
| Set q to point to 2 elements before the element p points to | q = p -2; |
| Use p to set the second to last element in the array to -5 | \*(p-1) = -5; |
| Let q point to the same element as p | q = p; |
| Print the address of p | printf(“%p\n”, &p); |
| Using p, print the number of elements separating \*p from the first element of a | printf(“%d\n”, p – a); |
| Write an if statement that prints “YES” if the value that p points to is smaller than the value that q points to and “NO” otherwise  if (\*p < \*q)  printf(“YES”);  else  printf(“NO”); | |

1. Write a recursive function that takes an array of integers and returns the number of even integers in it. (15 pts):

Int countEven(int array[ ], int n) {

if (n == 0)

return 0;

int count = 0;

if (array[n-1] % 2 == 0)

count++;

return count + countEven(array, n-1);

}

1. What is the output of the following c program (21 pts).

#include <stdio.h>

int a;

void addOne(int a) {

a++;

printf(“W. a = %d\n”, a);

}

int removeOne(int a) {

int b = a + 1;

printf(“R. b = %d\n”, b);

}

void swap(int \*a, int \*b) {

int temp = \*a;

\*a = \*b;

\*b = temp;

}

int main() {

a = 5;

int b = 20;

if (b > 15) {

int a = 53;

removeOne(b);

addOne(a);

printf(“X. a = %d\n”, a);

}

printf(“Y. a = %d, b = %d\n”, a, b);

swap(&a, &b);

printf(“Z. a = %d, b = %d\n”, a, b);

return 0;

}

R. b = 21

W. a = 54

X. a = 53

Y. a = 5, b = 20

Z. a = 20, b = 5

1. The function lowerTriangular takes as input a two dimensional array and its size and returns 1 if the array represents a lower triangular matrix and 0 otherwise. A two dimensional array is lower triangular if the number of columns is equal to the number of rows and all elements above the diagonal are zero. Finish the following program (29 pts)

/\* include any libraries needed \*/ (2)

#include<stdio.h>

#include<stdlib.h>

/\* declare the function lowerTriangular (prototype only) \*/ (3)

#define N 25

int lowerTriangular(int array[ ][N], int rows);

int main(){

/\* declare a 25x25 2-dimensional array \*/ (3)

int matrix[N][N];

/\* Initialize the arrays to random numbers \*/ (5)

int idx, idy;

for (idx = 0; idx < N; idx++) {

for (idy = 0; idy < N; idy++) {

matrix[idx][idy] = rand();

}

}

/\* Call lowerTriangular on the array and print the result \*/(5)

int result = lowerTriangular (matrix, N);

printf(“Result: %d\n”, result);

}

/\* define the function lowerTriangular \*/ (11)

int lowerTriangular(int array[ ][N], int rows) {

//Check if number of rows equals the number of columns

if (rows != N)

return 0;

int idx, idy;

for (idx = 0; idx < rows; idx++) {

for (idy = idx+1; idy < N; idy++) {

if (matrix[idx][idy] != 0) {

//Found non zero in top triangle

return 0;

}

}

}

//Checked all cells in top triangle and all were 0s

return 1;

}