

**Information and Computer Science Department**

**Summer Semester 143**

**ICS 103 – Computer Programming in C**

**Final Exam Key**

**Wednesday, August 12, 2015**

**Duration: 120 minutes**

**Question # 1 [20 points]**

Choose the most correct answer for each of the following:

1. Which of the following can be used to declare and initialize a 2-D array?
	1. **int x[][2] = {{1,2}, {3,4}};**
	2. int x[2][] = {{1,2}, {3,4}};
	3. int x[][] = {{1,2}, {3,4}};
	4. All of the above
2. Which of the following can be used to define a 2-D array as a parameter to a function prototype?
3. **int x[][2]**
4. int x[2][]
5. int x[][]
6. All of the above
7. Which of the following can be used to define a string as a parameter to a function prototype?
8. char str[]
9. char \*str
10. **All of the above**
11. Which of the following can be used to call the function **test** that has a 2-D array parameter?
12. **test(x)**
13. test(x[2][])
14. test(x[][2])
15. test(x[][])
16. Which of the following can be used to call the function **test** that has a string output parameter?
17. **test(x)**
18. test(x[])
19. test(&x)
20. Which of the following can be used to call the function **test** that has an integer output parameter?
21. test(x)
22. test(\*x)
23. **test(&x)**
24. The function call tolower('m')
	1. generates an error since 'm' is already in lower case
	2. returns the character 'M'
	3. **returns the character 'm'**
25. Consider the following array: int x[]={2, 4, 6, 7, 9, 13};

Using the linear search function, how many comparisons will be conducted if the target value is 7?

* 1. 3
	2. **4**
	3. 6
1. Consider the following array: int x[] = {20, 14, 6, 7, 9, 13};

What will be the content of the array after the first pass of selection sort?

* 1. {20, 14, 6, 7, 9, 13}
	2. **{6, 14, 20, 7, 9, 13}**
	3. {6, 7, 9, 13, 14, 20}
1. The result of **strcmp ("ace","an")** is:
	1. 0
	2. **-1**
	3. 1

**Question # 2 [20 points]**

Show the output of each program below.

| **Program** | **Output** |
| --- | --- |
| **#include <stdio.h>****int main (void)** **{**  **int k, i, x = 14, bin[5];** **for (k = 0; x != 0; k++)** **{** **bin[k] = x % 2;** **x /= 2;** **}** **for (i = k-1; i >= 0; i--)** **printf("%d", bin[i]);** **return (0);** **}** | 4 points**1110** |
| **#include <stdio.h>****#include <string.h>****void trim(char \*str);****int main (void)** **{**  **char str[10] = {'I','C','S',' ',' ','\0','1','0','3','\0'};** **trim(str);** **//** Show the content of str **return (0);** **}****void trim(char \*str)****{** **int n;** **for(n = strlen(str) - 1; str[n] == ' '; n = strlen(str) - 1)** **{** **str[n] = '\0';** **}****}** | 4 points

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **I** | **C** | **S** | **\0** | **\0** | **\0** | **1** | **0** | **3** | **\0** |

 |
| **#include <stdio.h>****int main(void)****{** **int i, j, x[3][3] = {{1,2,3}, {4,5,6}, {7,8,9}};** **for(i = 0; i < 3; i++)** **{** **for(j = 2; j >= i; j--)** **printf("%2d", x[j][i]);** **printf("\n") ;** **}** **return (0);** **}** | 6 points**7 4 1** **8 5** **9** |
| **#include <stdio.h>****int main (void)** **{** **int n = 10, \*m;** **m = &n;** **\*m = 20;** **printf("%d\n", n);** **return (0);** **}** | 2 points**20** |
| **#include <stdio.h>****void test(int n, int \*m);****int main (void)****{**  **int n = 10, m = 20;** **test(m, &n);** **printf("%d %d\n", n , m);** **return (0);** **}****void test(int n, int \*m)****{** **n = 100;** **\*m = 200;** **}** | 4 points**200 20** |

**Question # 3 [20 points]**

A 1D-array with distinct elements can be used to represent a set. Write an interactive C program that prompts for and reads two 1D-arrays each of size 5. It then does the following:

* It calls a function **setDifference** that returns the setDifference of the two arrays and the number of elements in the setDifference.
* It displays the setDifference.

**Note:**

* For two sets A and B, The set difference A – B = {$∀ x \in A and x \notin B\}$
* Your program must be general and it must use appropriate loops.
* Enter distinct elements in the two arrays and do not test whether the elements are distinct or not.

(i.e., assume that the user will enter distinct elements).

* The array reading must be done in the main and not in another function.
* The function setDifference must not contain calls to **scanf** and **printf** functions.

**Sample program runs:**

|  |  |
| --- | --- |
|  |  |

**#include <stdio.h>**

**#define SIZE 5**

**void setDifference(double array1[SIZE], double array2[SIZE], double difference[SIZE], int\* count);**

**int main(void)**

**{**

 **double array1[SIZE], array2[SIZE], difference[SIZE];**

 **int k, count1, count2;**

 **printf("Enter distinct elements of the first array:\n");**

 **for(k = 0; k < SIZE; k++)**

 **{**

 **printf("\tEnter element at index#%d: ", k);**

 **scanf("%lf", &array1[k]);**

 **}**

 **printf("\nEnter distinct elements of the second array:\n");**

 **for(k = 0; k < SIZE; k++)**

 **{**

 **printf("\tEnter element at index#%d: ", k);**

 **scanf("%lf", &array2[k]);**

 **}**

 **setDifference(array1, array2, difference, &count1);**

 **printf("\nThe difference is:\n");**

 **printf("{");**

 **for(k = 0; k < count1; k++)**

 **printf("%.2f ", difference[k]);**

 **printf("}");**

 **return 0;**

**}**

**void setDifference(double array1[SIZE], double array2[SIZE], double difference[SIZE], int\* count)**

**{**

 **int k, m, count1 = 0, index = 0;**

 **for(k = 0; k < SIZE; k++)**

 **{**

 **count1 = 0;**

 **for(m = 0; m < SIZE; m++)**

 **if(array1[k] != array2[m])**

 **count1++;**

 **if(count1 == SIZE)**

 **{**

 **difference[index] = array1[k];**

 **index++;**

 **}**

 **}**

 **\*count = index;**

**}**

**Question # 4 [15 points]**

Write a function **rowFrequencies** that receives an integer **target,** an integer 2D-array, number of rows, and number of columns. The function returns a 1D-array **frequencies**. The **frequencies** array contains the row frequencies of the target in the 2D-array.

For example if **target** is **7** and the 2D-array is:

**1 2 7**

**4 5 6**

**7 7 7**

**7 7 3**

Then the returned **frequencies** array is:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | 1 | 0 | 3 | 2 |

Note:

* Assume that the maximum number of columns NCOLS is defined as a constant
* Your function must be general, and must not contain **scanf** and **printf** statements.
* DO NOT WRITE THE main FUNCTION.

**void rowFrequencies(int target, int array[][NCOLS], int frequencies[], int nrows, int ncols)**

**{**

 **int k, n, counter;**

 **for(k = 0; k < nrows; k++)**

 **{**

 **counter = 0;**

 **for(n = 0; n < ncols; n++)**

 **if(array[k][n] == target)**

 **counter++;**

 **frequencies[k] = counter;**

 **}**

**}**

**Question # 5 [25 points]**

Write a complete C program that has the main function and another function called findstudent.

The main function reads data from a file called “input.txt”.

The first 2 values in this file represent the number of students and the number of quizzes. They are followed by lines of data equal to number of students. Each one of these lines contains the student id, his name, and the list of quiz marks obtained by that student.

The first step is to read this information into 3 arrays representing ids, names, and quizzes.

After reading the data, the program prompts the user for and reads a quiz number.

Then, it will call **findstudent** function which takes the 2D array quizzes, number of students, and the quiz number as input arguments and returns the row index corresponding to the student who got the highest mark in that quiz (assuming that there is one highest mark for each quiz).

After the call, the main function displays the highest mark of that quiz number along with the name of the student who obtained that mark and his id number.

The format of the output is as shown in the sample runs below.

Note: the number of students cannot exceed 100 and number of quizzes cannot excced 10

Sample input file “input.txt”



Sample program runs



#include <stdio.h>

#include <stdlib.h>

#define NSTUD 100

#define NQUIZS 10

int findstudent(double q[][NQUIZS], int nstd, int quiznb);

int main(void)

{

 int ids[NSTUD], i , j, nbstud, nbquizs, quiznb, index;

 double qz[NSTUD][NQUIZS];

 char names[NSTUD][31];

 FILE \*in;

 in = fopen("input.txt","r");

 if(in == NULL)

 {

 printf("No input file found");

 exit(1);

 }

 fscanf(in, "%d%d", &nbstud, &nbquizs);

 for(i = 0; i < nbstud; i++)

 {

 fscanf(in,"%d%s", &ids[i], names[i]);

 for(j = 0; j < nbquizs; j++)

 fscanf(in, "%lf", &qz[i][j]);

 }

 printf("\nEnter quiz number >");

 scanf("%d", &quiznb);

 index = findstudent(qz, nbstud, quiznb);

 printf("%.2f is the highest mark in quiz %d\n", qz[index][quiznb-1], quiznb);

 printf("obtained by %s with id %d\n", names[index], ids[index]);

 return 0;

}

int findstudent(double qz[][NQUIZS], int nbstud, int quiznb)

{

 int i, j, maxpos = 0;

 for(i = 1; i < nbstud; i++)

 if(qz[i][quiznb-1] > qz[maxpos][quiznb-1])

 maxpos=i;

 return maxpos;

}