**Information and Computer Science Department**

**ICS 103: Computer Programming in C**

**Final Exam, Term 171**

**Sunday, January 7, 2018**

**Duration: 120 minutes**

**Name: SOLUTION KEY**

**Instructor and Section: Select one**

|  |  |
| --- | --- |
| **Instructor** | **Section** |
| Dr. Tareq El-Bassuny  |  [ ] **01** (UT 7 – 7:50) [ ] **02** (UT 8 – 8:50) [ ] **03** (UT 9 – 9:50) |
| Dr. Mohamed Balah |  [ ] **04** (UT 10 – 10:50) [ ] **05** (UT 11 – 11:50) [ ] **21** (UT 13 – 13:50) |
| Dr. Md Rafiul Hassan |  [ ] **06** (UT 12 – 12:50) [ ] **07** (UT 13 – 13:50) |
| Mr. Muhammad Aslam |  [ ] **08** (MW 7 – 7:50) |
| Dr. El-Sayed El-Alfy |  [ ] **09** (MW 8 – 8:50) [ ] **10** (MW 9 – 9:50) |
| Dr. Muhamed Mudawar |  [ ] **11** (MW 10 – 10:50) [ ] **12** (MW 11 – 11:50) |
| Dr. Ali Al-Suwaiyan |  [ ] **13** (MW 12 – 12:50) [ ] **19** (UT 11 – 11:50) [ ] **20** (UT 12 – 12:50) |
| Dr. Ayman Hroub |  [ ] **14** (MW 13 – 13:50) |
| Dr. Uthman Baroudi |  [ ] **15** (UT 7 – 7:50) [ ] **16** (UT 8 – 8:50) |
| Dr. Louai Al-Awami |  [ ] **17** (UT 9 – 9:50) [ ] **18** (UT 10 – 10:50) |
| Dr. Kamal C |  [ ] **22** (MW 7 – 7:50) [ ] **23** (MW 8 – 8:50) [ ] **25** (MW 10 – 10:50) |
| Dr. Samer Arafat |  [ ] **24** (MW 9 – 9:50) [ ] **26** (MW 11 – 11:50) |

**Instructions**:

1. Answer all questions. Make sure your answers are **clear** and **readable**.
2. The exam is closed book and closed notes. No calculators or any helping aides are allowed. Make sure to turn off your mobile phone and keep it in your pocket.

3. If there is no space on the front of the page, use the back of the page. Indicate this clearly.

|  |  |  |  |
| --- | --- | --- | --- |
| **Question** | **Maximum Points** | **Earned Points** | **Remarks** |
| **1** | **30** |  |  |
| **2** | **10** |  |  |
| **3** | **15** |  |  |
| **4** | **17** |  |  |
| **5** | **12** |  |  |
| **6** | **16** |  |  |
| **Total** | **100** |  |  |

**Question 1**

 [ **30 points**]

|  |  |
| --- | --- |
| **Code/Code fragment** | **Your answer** |
| #include <stdio.h> // 4 points int main(void) {int j,i;int a[3][3] = {0} ;for (i=0; i<3 ; i++)  for(j=0; i>j ; j++)  a[i][j] = i+j ; for (i=2; i<3; i++)  for(j=0; j<3; j++) printf("%d\n",a[i][j]);return 0;} | **2****3****0** |
| #include<stdio.h> // 4 pointsint get(int c, int \*d);int main() { int \*p,a=2,c=2,d=8, value=0; p=&a;  \*p = c+1; a = d+1; printf("%d ",a); value = get(d,&c); printf("%d\t%d\n", c, d); return 0; } int get(int d, int \*c) { \*c = \*c +2; d = d+5; return 7; } | **9 4 8** |
| #include <stdio.h> //5 points#include <string.h>int main() { char input[] = "A+B-C\*D/E"; char delims[] = "/\*"; char \*token;   token = strtok( input, delims);  int n = 0; while ( token != NULL ) { puts(token) ; n++; token = strtok(NULL, delims );  } printf("%d", n) ; return 0;} | **A+B-C****D****E****3** |
| #include<stdio.h> //5 pointsint changevalue1(int x) ;int changevalue2(int \*x) ; int main(){     int y=10, z=15;        changevalue1(y);     changevalue2(&z);     printf("%d\n",y);    printf("%d\n",z);    changevalue2(&y); changevalue1(z);     printf("%d\n",y);     printf("%d\n",z); return 0;}int changevalue2(int \*x) {     return(\*x += 5); }int changevalue1(int x) {     return(++x); } | **10****20****15****20** |
| #include <stdio.h> // 4 pointsint main(void) {int j=0,i;while (j<=2) { for(i=2; i>j ; i--)  if ((i+j)<1) printf("%d ",j); printf("%d\n",i); j++;}return 0;} | **0****1****2** |
| #include <stdio.h> //4 pointsint main(){  int x=5, y=8;  x = y++ ; printf("%d %d\n",x,y); x++;  printf("%d %d\n",x,y); y= ++x;  printf("%d %d\n",x,y);  return 0;} | **8 9****9 9****10 10** |
| #include<stdio.h> //2 points#include<string.h>int main(void){ int i, found=0; char pw[10]; char users[3][10] = {"Admin","User 1", "User 2"}; char password[3][10] = {"ics 103", "1234", "abc"}; printf("Enter your password:"); scanf("%s",pw); for(i=0;i<3 && !found; i++){ if(!strcmp(pw,password[i])){ printf("Hello, %s",users[i]); found=1; } } if(!found) printf("Wrong password!"); return 0;} | **User Input:****Enter your password:** ics 103 |
| **Wrong password!** |
| #include<stdio.h> //2 pointsint main(void){ int n, i=0; char in[10], word[6];  gets(in);  for(n=7; n<=15 ; n+= 2){ word[n%5] = in[i]; i++; }  word[5]='\0'; puts(word);  return 0;} | **User Input:**Email |
| **laEim** |

**Question 2**  [ **10 points** ]

1. Convert the following if-statements into their equivalent nested switch-statements. Assume that the variables **k** and **m** are integer variables and that the variable **x** is initialized.

[**6 points**]

|  |  |
| --- | --- |
| if (k == 11) x += 11%3;else if (k == 8|| k == 4) x -= k;else { if (m == 1) x = 0; else if(m == 3) x \*= 2;} | switch(k){ // 1 point case 11: x += 11%3; break; // 1 point case 4: // -1 point if missing case 8: x -= k; break; // 1 point default: switch(m){ // 1 point case 1: x = 0; break; // 1 point case 3: x \*= 2; break; // 1 point } } |

1. Use the Selection sort algorithm in sorting the array **1, 11, 10, 2, 1, 5, 12** in increasing order (i.e., from lowest to highest) by showing the contents of the array at the end of each of the first two passes. The top element should be the number with the least value in the array.

[ **4 points**]

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  |  | Pass1 |  | Pass2 |  |
| 1 |  | **1** |  | 1 |  |
| 11 |  | 11 |  | **1** |  |
| 10 |  | 10 |  | 10 |  |
| 2 |  | 2 |  | **2** |  |
| 1 |  | 1 |  | 11 |  |
| 5 |  | **5** |  | 5 |  |
| 12 |  | 12 |  | 12 |  |

**Question 3**  [ **15 points** ]

1. Write down a function in C programming which solves the following problem:

A point $\left(x\_{p},y\_{p}\right)$ lies inside a circle that has the center point $(x\_{c},y\_{c})$ and radius $r$, if the following relation is satisfied:

$$\left(x\_{p}-x\_{c}\right)^{2}+\left(y\_{p}-y\_{c}\right)^{2}<r^{2}$$

 Your function input parameters are: $x\_{p}, y\_{p}, x\_{c}, y\_{c}, r$ of type double

 Your function will return 1 if the point $\left(x\_{p}, y\_{p}\right)$ lies inside the circle and 0 if the point does not lie inside the circle. **[5 points]**

int inside\_circle (double xp, double yp, double xc, double yc, double r)

{

 return ((xp-xc)\*(xp-xc) + (yp-yc)\*(yp-yc) < (r\*r));

}

2 points for the header 3 points for the return statement

// alternate solution

int inside\_circle (double xp, double yp, double xc, double yc, double r) // 2 points

{

if(((xp-xc)\*(xp-xc) +(yp-yc)\*(yp-yc)<(r\*r)) // 1 point

 retrun 1; // 1 point

else

 return 0; // 1 point

}

1. Consider the following function:

void test(int n){

 if ( n%2 == 0 ) printf (“2”);

 else if ( n%3 == 0 ) printf (“3”);

 else if ( n%5 == 0 ) printf (“5”);

 else if ( n%7 == 0 ) printf (“7”);

 else printf(“No divisors!”);

 }

Rewrite the above function below without using the keywords “***else***” or “***switch***”. You also cannot use the keyword “***if***” more than once in your code. [Hint: Store the numbers 2,3,5,7 in an array and process this array in a loop and exit the function if required] **[10 points]**

void test (int n){ // 1 point

 int x[] = {2,3,5,7},i; // 2 points for array+values

 // 1 point for array and no values

 for(i=0;i<4;i++) // 1 point

 if(n%x[i]==0){ // 2 points (-1 for missing **{}**)

 printf(“%d”,x[i]); // 1 point

 **return;** // 2 points for control

// ignore error if return 0 or 1

 } // 0 points if exit() is used

 printf(“No divisors!”); // 1 point

}

void test (int n){ // 1 point

 int x[] = {2,3,5,7},i; // 2 points for array+values

 for(i=0;i<4;i++) // 1 point

 if(n%x[i]==0){ // 2 points (-1 for missing **{}**)

 printf(“%d”,x[i]); // 1 point

 **break;** // 1 point for break

 }

 **if (i==4)** printf(“No divisors!”);// 1 point(0 for **missing if**)

}

The problem with this solution is that it uses 2nd **if** statement outside **for** loop, which is against the problem statement. M**aximum of** **9 points**.

A third solution is to use a **while** loop and a Boolean variable for early loop termination. However, a 2nd **if** statement is needed outside the loop as in the second solution that uses **break**. M**aximum of** **9 points**.

**Question 4** [**17 points**]

1. Matrix transpose is a linear algebra operator in which the matrix rows ae switched into columns and vice versa.

For Example,

**3 5 -1**

**4 90 20**

**A =**

**3 4**

**5 90**

**-1 20**

**B = Transpose(A)=**

Write a C function that takes a two-dimensional matrix ***A*** of integers (the maximum number of rows and columns are 8x8) and its dimensions as input parameters that are passed from the main program, and then it finds the transpose of ***A*** and returns it through another matrix ***B***. (Note: the matrix ***B*** is an output parameter for to the function). [**9 points**]

void Transpose(int A[][8],int B[][8],int m,int n)

//3 points (2 points for two matrices + 1 point for header)

{

 int i, j; // 1 point

 for (i = 0; i < m; i++) // 1 point

 for(j = 0 ; j < n ; j++ ) // 2 points

 B[j][i] = A[i][j]; // 2 points

}

1. Write a function that receives an array named ***list***[] consisting of **n** integer values. The function should count and return the number of pair of values that are not in ascending order. For example, if list[] = {1, 6, 5, 7, 9, 4, 3} then the list has following successive pairs: {1,6}, {6,5}, {5,7}, {7,9}, {9,4}, {4,3}. Therefore, there are 3 pairs of values that are not in ascending order in the list of successive pairs: {6,5}, {9,4}, and {4,3}, and the function should return 3 in this exampe. If the list is completely sorted then the function should return 0. [ **8 points** ]

int test(int x[], int size){ // 1 point

 int i, count=0; // 1 point

for (i=0;i<size-1;i++) // 2 point

 if(x[i]>x[i+1]) // 2 point

 count++; // 1 point

return count; // 1 point

}

**Question 5** [ **12 points** ]

Write a function to find a student’s name (either first or last name) from a data file “***mydata.txt***”. The data file “***mydata.txt***” stores the following data (each name is represented as: first name last name, e.g., Muhammad Abdul, here the first name is Muhammad and last name is Abdul). If the name (either first name or last name) is found in the data file, the function returns **1** else the function returns **-1**. The input to your function should be: student name (either first or last name) (must be all small letters) that you are searching for. Your function must open the data file inside the function and check for possible file open errors. Consider the maximum size of a string is 200.



/\* function header and variable declarations **3 points**. If the input parameter is not correct, - 1 point. If return type is not integer - 1 point. If any of the required variable declaration is missing -0.5 point for each missing variable declaration \*/

int findname(char item[]){ //1 point

 FILE \*in; //1 point

 char str[200]; //1 point

 int i;

 /\* file open for reading purpose and check fill opening error **2 points** \*/

 in = fopen("mydata.txt","r"); //1 point

 if(in==NULL){

 printf("File mydata.txt not found!"); //1 point

 **return -1;**

 }

 /\* loop for reading data from file **2 points**. Consider the following loop is also correct if in place of fgets() function, fscanf() function is used, e.g.,s while((fscanf(in,"%s",str))!=EOF) \*/

 while((fgets(str,200,in))!=NULL){ //2 points

/\* convert item to upper case or convert str to lower case **2 points** \*/

 for(i=0;i<strlen(str);i++){ //1 point

 str[i]= tolower(str[i]); //1 point

 }

 /\* compare the input data with the data read from file using strcmp (if fscanf was used) or strstr (if fgets was used) and return statemnet

 if((strstr(str,item))!=NULL) // 2 points

 return 1;

 }

 return -1; // **1 point**

}

// alternate solutions may declare \*status and then uses status with fgets(…)

// instead of using fgets(…) inside the while header

**Question 6** [**16 points**]

Write a program that reads a 2D array **F** (shown below) with size = 4x5 elements that contains arbitrary values from 0 to 7. The program computes and prints the co-occurrence matrix **G**. Each element in **G** at row index x and column index y counts the frequency of having x followed by y in the same row of matrix **F**. For example, **G**[0][1] has the value 2 indicating that the sequence 0, 1 occurred twice in matrix **F**, as shown below by the 2 arrows. Your program **should not check** whether the values are outside of the 0 to 7 range, or not.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | **0** | **4** | **5** | **7** |  | **1** | **2** | **0** | **0** | **1** | **0** | **0** | **0** |
| **1** | **2** | **4** | **6** | **0** | **0** | **0** | **1** | **0** | **1** | **0** | **0** | **0** |
| **3** | **4** | **6** | **0** | **1** | **0** | **0** | **0** | **0** | **1** | **0** | **0** | **0** |
| **7** | **4** | **0** | **1** | **4** | **0** | **0** | **0** | **0** | **1** | **0** | **0** | **0** |
| **F** | **1** | **0** | **0** | **0** | **0** | **1** | **2** | **0** |
| **0** | **0** | **0** | **0** | **0** | **0** | **0** | **1** |
| **2** | **0** | **0** | **0** | **0** | **0** | **0** | **0** |
| **0** | **0** | **0** | **0** | **1** | **0** | **0** | **0** |
| **G** |

Sample input and its output:

|  |
| --- |
| Enter the values for matrix F (4x5):0 0 4 5 71 2 4 6 03 4 6 0 17 4 0 1 4The matrix G is as follows::1 2 0 0 1 0 0 00 0 1 0 1 0 0 00 0 0 0 1 0 0 00 0 0 0 1 0 0 01 0 0 0 0 1 2 00 0 0 0 0 0 0 12 0 0 0 0 0 0 00 0 0 0 1 0 0 0 |

#include<stdio.h>

int main(){

 /\* 2 points for correct variable declaration (- 1 point for not initializing the array. If inside the program the respective cells are assigned to zero then do no problem \*/

 int x[4][5],y[8][8] ={0},i,j,k,m;

 printf("Enter the values for matrix F (4x5):\n"); //1

 //point

 for(i=0;i<4;i++) //1 point

 for(j=0;j<5;j++) //1 point

 scanf("%d",&x[i][j]); //1 point

 for(i=0;i<4;i++) //1 point

 for(j=0;j<4;j++){ //1 point

 k = x[i][j]; //1 point

 m = x[i][j+1]; // 2 points //5 pts

 y[k][m]=y[k][m]+1; // 2 points

 }

// alternate solution for last segment with 5 points

// y[x[i][j]][ x[i][j+1]] +=1 ; // statement gets 5 pts

 printf("\nThe matrix G is as follows:: \n");

 for(i=0;i<8;i++){ // 1 point

 for(j=0;j<8;j++) // 1 point

 printf("%d\t",y[i][j]); //1 point

 printf("\n");

 }

 return 0;

 }