**Information and Computer Science Department**

**ICS 103: Computer Programming in C**

**Final Exam, Term 173**

**Monday, August 13, 2018**

**Duration: 120 minutes**

**Name: ID:**

**Instructor and Section: Select one**

|  |  |
| --- | --- |
| **Instructor** | **Section** |
| Dr. Mahmoud Niazi | [ ] **01** (08:10 – 09:00) [ ] **02** (09:20 – 10:10) [ ] **03** (10:30 – 11:20) |
| Dr. Farag Azzedin | [ ] **04** (08:10 – 09:00) [ ] **05** (09:20 – 10:10) [ ] **06** (10:30 – 11:20) |

**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** | **40** |  |  |
| **2** | **15** |  |  |
| **3** | **14** |  |  |
| **4** | **14** |  |  |
| **5** | **17** |  |  |
| **Total** | **100** |  |  |

**Question 1: 40 points**

What is the output of the following code fragments? [40 points]

|  |  |
| --- | --- |
| Code fragment | Output |
| #include<stdio.h> // 2 pts  int main(){  char s[] = "Fine";  \*s = '\0';  printf("%s", s);  } | **Nothing will be printed**  **2 points or 0** |
| #include<stdio.h> // 2 pts  int main(){  int i,j,k,count=0;  for(i = 1; i <= 5; i++)  count++;  for(j = 10; j <= 12; j++)  for (k = 7; k >=6; k--)  count++;  printf("%d\n", count);  } | **11**  2 points or 0 |
| #include<stdio.h> // 2 pts  int main(){  int i = 0;  for (++i; i == 1; i = 2)  printf("Inside for loop ");  return(0);  printf("Outside loop\n");  } | **Inside for loop**  **2 points or 0** |
| #include<stdio.h> // 2 pts  int main() {  int i, j;  int x[2][2]={{2,3},{4}};  for(j=1;j>=0;j--)  for(i=0;i<=1;i++)  printf("%d ",x[i][j]);  return (0);  } | **3 0 2 4**  **0.5 point each value** |
| #include<stdio.h> // 2 pts  int main() {  int A[5] = {2, 0, 3, 4};  printf("%d %d", A[A[4]]+2, A[A[2]-1]);  return (0);  } | **4 3**  **1 point each value** |
| #include<stdio.h> //4 pts  int main() {  int arry[][2] = {{5, 12},  {9, 6}, {3, 7}}, k, m;  for(m = 1; m >= 0; m--){  for(k = 1; k >= 0; k--){  printf("%d ", arry[2 - k][1 - m]);  }  printf("\n");  }  return (0);  } | **9 3**  **6 7**  1 point each value |
| #include <stdio.h> // 3 pts  void test (int \*b, int a, int i);  int main() {  int y[]= {1,2,3},i;  for(i=1;i<=2;i++)  test(&y[i-1],y[i],i);  for(i=0;i<3;i++)  printf("%d\t",y[i]);  return 0;  }  void test (int \*b, int a, int i) {  \*b=a+i;  a=a\*2;  } | **3 5 3**  **1 point each value** |
| #include <stdio.h> // 2 pts  int main() {  int i, j, \*ptr, \*ptr1;  i = j = 0;  ptr = &i;  ptr1 = &j;  if(ptr == ptr1)  {  printf("True");  }  else  {  printf("False");  }  return 0;  } | **False**  **2 points or 0** |
| What is the output if the input file contains:  **ICS 103 is a programming course**  #include <stdio.h> // 2 pts  int main() {  FILE \*inptr;  inptr = fopen("input.txt", "r");  char str1[128], str2[128];  fgets(str1, 6, inptr);  fscanf(inptr, "%s", str2);  puts(str1);  printf("%s", str2);  return 0;  } | **ICS 1**  **03**  **1 point each line** |
| #include<stdio.h> // 4 pts  #include<string.h>  int main(void){  int i;  char users[10][50], \*token;  char data[50] = "Admin\*User1@\*User2\*%@";  char delims[] = "\*@U%" ;  token = strtok(data, delims);  i = 0;  while(token != NULL){  strcpy(users[i], token) ;  puts(users[i]) ;  i++;  token = strtok(NULL,delims);  }  printf("%d", i) ;  return 0;  } | **Admin**  **ser1**  **ser2**  **3**  **1 point each line** |
| #include<stdio.h> //2 pts  #include<string.h>  int main(){  char s[] = "HelloHi";  s[2] = '\0';  printf("%d %s", strlen(s) + 2,s);  } | **4 He**  **1 point each** |
| #include <stdio.h> // 3 pts  int fun(int a, int\* b);  int main(void) {  int x[] = {1, 2, 3, 4};  printf("%d %d\n",fun(x[1], &x[3]));  printf("%d %d\n", x[1], x[3]);  return 0;  }  int fun(int a, int\* b) {  \*b = 3 \* \*b;  a = 2;  return a + \*b;  } | **14**  **2 12**    **1 point each value** |
| #include<stdio.h> // 4 pts  #include <string.h>  int main(void){  char string1[80] = "KFUPM";  char string2[80] = "ICS";  char temp[80]="DHAHRAN";  strcpy(temp, string1);  puts(temp);  printf("%d\n",strcmp(temp,string1));  strcat(string2,string1);  puts(string2);  if (strstr(string2,"SK")!=NULL)  printf("True\n");  else  printf("False\n");  return 0;  } | **KFUPM**  **0**  **ICSKFUPM**  **True**  **1 point each line** |
| #include<stdio.h> // 4 pts  int main(void){  int i,j,rows = 2,cols=2;  int a[rows][cols];  for(i=0; i<rows; i++) {  for(j=0; j<cols; j++)  scanf("%d", &a[cols-j-1][rows-i-1]);  }  for(i=0; i<rows; i++)  for(j=0; j<cols; j++)  printf("%d ", a[i][j]);  return 0;  }  Values entered by the user are  **1 2 3 4** | **4 2 3 1**  **1 point each value** |

|  |  |
| --- | --- |
| //2 pts  int x[] = {3, 4, 0, 1, 2, 25, 15};  Selection sort algorithm is applied to sort array *x* in a decreasing order (i.e., from highest to lowest). Write on the right side array *x* after the first pass of the selection sort algorithm. | **{25, 4, 0, 1, 2, 3, 15}**  **Or**  **{25, 4, 15, 1, 2, 3, 0}**  **If not one of the 2 solutions, 0** |

**Question # 2 [15 points]**

A text-file **input.txt** contains only English words where each word is separated from the next word by one or more of the following delimiters: white-space characters: blank, tab, or new-line character, and punctuation characters: **, . ; : ? !**

Write a C program that prompts for and reads a letter from the user. The program then displays on the screen the number of words present in the file and starting with this letter.

Sample input.txt

|  |
| --- |
| this is a file  file is there but it is empty  this is the key for this file  bye |

Sample program runs for the above input file:

|  |  |
| --- | --- |
| **Enter a letter: b**  **Words starting with b = 2** | **Enter a letter: m**  **Words starting with m = 0** |
| **Enter a letter: e**  **Words starting with e = 1** | **Enter a letter: f**  **Words starting with f = 4** |

**Note:**

* Assume that the input file **input.txt** exists
* Assume that there is a maximum of 80 characters in a line of **input.txt**
* Assume that the input word has a maximum size of 30.
* Your program must be general; it must work for any input file with a maximum of 80 characters per line.
* The output of your program must be similar to the given sample program runs.

**#include <stdio.h>**

**#include <string.h>**

**int main(void){**

**FILE \*infile;**

**char letter;**

**char line[81], keyWord[31], delimeters[] = " \t\n?.,;!:", \*word;**

**int count = 0;**

**infile = fopen("input.txt", "r");**

**printf("Enter a letter: ");**

**scanf("%c", &letter);**

**while (fgets(line, 80, infile) != NULL) {**

**word = strtok(line, delimeters);**

**while (word != NULL) {**

**if ( word[0] == letter ) {**

**count++;**

**}**

**word = strtok(NULL, delimeters);**

**}**

**}**

**printf("Words starting with %c appears %d times\n", letter, count);**

**fclose(infile);**

**return 0; }**

**Question 3 : [14 points]**

Write a C function SplitArray that takes one-dimensional array of integer values and its size. It returns **two one-dimensional arrays**. The first array will contain the odd integers present in the input array. The second array will contain the even ones. The function must also return how many odd and even numbers found.

For example if the input array is: { 5,7,9,8,14,20,55,13}

The 2 output arrays will contain the values: {5,7,9,55,13} and {8,14,20}. The function will return 5 (number of odd values) and 3 (number of even values)

Note: Write the function definition only. Do not write the main function

void SplitArray(int x[], int o [],int e [],int size, int \*even, int \*odd)

{

\*even=0;

\*odd=0;

for(int i = 0; i < size; i++)

if (x[i] % 2 == 0) {

e[\*even] = x[i];

\*even = \*even + 1;

}

else {

o[\*odd] = x[i];

\*odd = \*odd + 1;

}

}

**Question 4 : [14 points]**

Write an integer function **insertRows** that has the following **four** parameters:

* Two 2D-arrays **x** and **y** of type **double** with the same dimensions.
* The number of rows and the number of columns in the 2D-arrays.

The function **insertRows** inserts the rows of array **y** into the array **x,** starting from the first empty row of array **x**. Your function returns **0** without modifying array **x** if the number of empty rows in array **x** is less than the number of rows in array **y**; otherwise it modifies array **x** and then returns **1**.

**Example:** in this example MAXROWS=9 , nbrows=4 for both arrays.

If array **x** is: array **y** is:

|  |  |  |
| --- | --- | --- |
| 1.0 | 5.0 | 9.0 |
| 2.0 | 6.0 | 10.0 |
| 3.0 | 7.0 | 11.0 |
| 4.0 | 8.0 | 12.0 |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

|  |  |  |
| --- | --- | --- |
| 13.0 | 11.0 | 30.0 |
| 23.0 | 22.0 | 80.0 |
| 35.0 | 33.0 | 12.0 |
| 43.0 | 44.0 | 40.0 |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

Then, the modified array **x** is:

|  |  |  |
| --- | --- | --- |
| 1.0 | 5.0 | 9.0 |
| 2.0 | 6.0 | 10.0 |
| 3.0 | 7.0 | 11.0 |
| 4.0 | 8.0 | 12.0 |
| 13.0 | 11.0 | 30.0 |
| 23.0 | 22.0 | 80.0 |
| 35.0 | 33.0 | 12.0 |
| 43.0 | 44.0 | 40.0 |
|  |  |  |
|  |  |  |

**Note:**

* Assume a constant **MAXROWS** that stores the maximum number of rows in array **x** and **y** is defined outside all functions. Same for **MAXCOLS**
* Your function must be general, it must not contain **scanf** and **printf** calls, and it must use appropriate **nested-loop**.

**int insertRows(double x[][MAXCOLS], double y[][MAXCOLS], int nrows, int ncols){**

**int k, r, c;**

**if(nrows > MAXROWS - nrows)**

**return 0;**

**r = nrows;**

**for(c = 0; c <= nrows - 1; c++){**

**for(k = 0; k <= ncols - 1; k++){**

**x[r][k] = y[c][k];**

**}**

**r++;**

**}**

**return 1;**

**}**

**Version2**

**int insertRows(double x[][MAXCOLS], double y[][MAXCOLS], int nrows, int ncols){**

**int k, r, c;**

**if(nrows > MAXROWS - nrows)**

**return 0;**

**for(c = 0; c < nrows; c++){ //can fill row-wise or column-wise**

**for(k = 0; k < ncols ; k++){**

**x[c+nrows][k] = y[c][k];**

**}**

**}**

**return 1;**

**}**

**Version 3**

**int insertRows(double x[][MAXCOLS], double y[][MAXCOLS], int nrows, int ncols){**

**int k, r, c;**

**if(nrows > MAXROWS - nrows)**

**return 0;**

**for(k = 0; k <= ncols - 1; k++){ // filling column-wise**

**r=nrows;**

**for(c = 0; c <= nrows - 1; c++){**

**x[r][k] = y[c][k];**

**r++;**

**}**

**}**

**return 1;**

**}**

**Question 5 : [18 points]**

Write a C program that prompts the user to enter the number of rows and columns of a 2-D array. The program will create the 2-D array with the sizes input by the user. Then, it will read the values entered by the user row-wise to initialize the 2-D array. The values are integer.

Your program should find and display on the screen:

* The row index with the maximum sum.
* The maximum row sum.

Sample run

|  |
| --- |
| Please enter # of rows and # of cols: 4 3  Enter values of 4x3 array row-wise  2 4 20  -7 10 35  15 50 10  4 7 45  Maximum row sum is 75  Maximum sum was found at row index 2 |

**#include <stdio.h>**

**int main (void){**

**int i, j, rows, cols,row\_sum,max\_sum,max\_index;**

**printf("Please enter # of rows and # of cols: ");**

**scanf("%d%d", &rows, &cols);**

**int a[rows][cols];**

**printf("Enter values of %dx%d row-wise \n", rows, cols);**

**for (i = 0; i < rows; i++)**

**for (j = 0; j < cols; j++)**

**scanf("%d", &a[i][j]);**

**max\_sum = 0;**

**max\_index = 0;**

**for(j=0;j<cols;j++) // init*alize max\_sum with sum of row 0***

**max\_sum=max\_sum+a[0][j];**

**for (i = 1; i < rows; i++) {**

**row\_sum = 0;**

**for (j = 0; j < cols; j++)**

**row\_sum = row\_sum + a[i][j]; // add values of row i**

**if ( row\_sum > max\_sum) {**

**max\_sum = row\_sum;**

**max\_index = i;**

**}**

**}**

**printf("Maximum row sum is %8.2f\n", max\_sum);**

**printf("Maximum sum was found at row index %d\n", max\_index);**

**return(0);}**