Information and Computer Science Department

Second Semester 162

ICS 103 – Computer Programming in C

Final Examination

Tuesday, 23rd May 2017

Duration: 120 minutes

|  |  |  |  |
| --- | --- | --- | --- |
| Name: | KEY | Serial # |  |

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| ID#: |  |  |  |  |  |  |  |  |  | Section# |  |

**Instructions:**

1. Answer all questions. Make sure your answers are clear and readable.
2. Make sure there are **5** questions in **11** pages.
3. 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.

1. If there is no space on the front of a question’s page, use the back of the page. Indicate this clearly.

|  |  |  |  |
| --- | --- | --- | --- |
| Question # | Maximum Grade | Obtained Grade | Remarks |
| 1 | 15 |  |  |
| 2 | 35 |  |  |
| 3 | 15 |  |  |
| 4 | 15 |  |  |
| 5 | 20 |  |  |
| Total | 100 |  |  |

**Question # 1 [15 points]**

1. [5 points] Convert the following C if-statement into equivalent switch-statements. Assume that the variables **k** and **m** are integer variables and that the variable **x** is initialized.

|  |  |
| --- | --- |
| if(k == 6)  x += 3;  else if (k == 3|| k == 5)  x -= 3;  else {  if(m == 2)  x /= 5;  else if(m == 8)  x \*= 2;  } | switch(k){  case 6: x += 3; break;  case 3:  case 5: x -= 3; break;  default: switch(m){  case 2: x /= 5; break;  case 8: x \*= 2; break;  }  } |

1. [5 points] Convert the following program fragment into an equivalent fragment that uses **nested for-loops** instead of **nested while-loops**:

|  |  |
| --- | --- |
| double sum = 0;  int i = 12, k;  while(i > 1){  k = i;  while(k <= 20){  sum = sum + i + k;  k++;  }  printf("sum = %0.1f\n",sum);  i = i - 2;  } | double sum = 0 ;  int i, k;  for(i = 12; i > 1; i = i - 2){  for(k = i; k <= 20; k++)  sum = sum + i + k;  printf("sum = %0.1f\n",sum);  } |

1. [5 points] Trace the Selection sort algorithm in sorting the array **8, 4, 60, 1, 2, 25, 15** in decreasing order (i.e., from highest to lowest) by showing the contents of the array at the end of each of the first two passes:

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | Pass1 |  | Pass2 |  |  |  | Pass1 |  | Pass2 |  |
| 8 |  | **60** |  | 60 |  | 8 |  | 8 |  | **8** |  |
| 4 |  | 4 |  | **25** |  | 4 |  | 4 |  | 4 |  |
| 60 |  | 8 |  | 8 | or | 60 |  | 60 |  | 60 |  |
| 1 |  | 1 |  | 1 |  | 1 |  | 15 |  | 15 |  |
| 2 |  | 2 |  | 2 |  | 2 |  | 2 |  | 25 |  |
| 25 |  | 25 |  | 4 |  | 25 |  | 25 |  | **2** |  |
| 15 |  | 15 |  | 15 |  | 15 |  | **1** |  | 1 |  |

**Question # 2 [35 points]**

Write the output of each of the following C programs or program fragments:

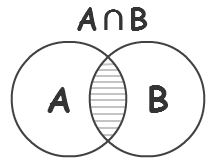
|  |  |
| --- | --- |
| [3 points]  What is the output if the input file is:  **ICS 243 FINAL EXAM**  FILE \*inptr;  inptr = fopen("input.txt", "r");  char str1[128], str2[128];  fscanf(inptr, "%s", str2);  fgets(str1, 4, inptr);  puts(str1);  printf("%s", str1);  printf("%s", str2); | **Output:**  24  24ICS |
| [5 points]  int x[ ][4] = {{2, 0, 1, 3}, {4, 1, 0, 3},{3, 0, 3, 4}}, r, c;  int y[5] = {0};    for(r = 0; r <= 2; r++){  for(c = 0; c <= 3; c++){  y[x[r][c]] ++;  }  }    for(c = 4; c >= 0; c--)  printf("%d ", y[c]); | **Output:**  2 4 1 2 3 |
| [5 points]  int x[] = {3, 4, 1, 2, 6, 5};  int k;  for(k = 2; k <= 4; k++){  x[k - 2] = x[k - 2] + x[k + 1];  x[k] = x[k - 1];  }  for(k = 0;k <= 4; k++)  printf("%d ", x[k]); | **Output:**  5 10 9 4 4 |
| [5 points]  #include <stdio.h>  int funcn(int a, int b, int\* c, int\* d);  int main(void) {  int a = 4,y = 3, x[] = {2, 5};  printf("%d ",funcn(a, x[1], &x[0], &y));  printf("%d %d %d %d",a, x[0], x[1], y);  return 0;  }  int funcn(int a, int b, int\* c, int\* d) {  a = 3 \* b;  b = 5;  \*c = 2;  \*d = 4 + \*c ;  return a + b + \*d;  } | **Output:**  26 4 2 5 6 |
| [7 points]  #include <stdio.h>  #include <string.h>  #include <ctype.h>  int main(void){  char string1[80] = "APPLES";  char string2[80] = "ORES";  int k, m;  char temp[80];  strcpy(temp, string1);  strcpy(string1, string2);  strcpy(string2, temp);  puts(string1);  puts(string2);  printf("\n");  for(m = strlen(string1) - 1; m >= 1; m--){  string2[m] = tolower(string1[m]);  printf("%c", string1[m - 1]);  }  puts(string2);  return 0;  } | **Output:**  ORES  APPLES    EROAresES |
| [5 points]  #include <stdio.h>  #include <ctype.h>  #include <string.h>  int main(void){  //Note: There is one space between H and 3  // and one space between 9 and $ in string str  char str[ ] = "BH 369 $Ac)Rd";  int k = 0;  while(str[k] != '\0'){  if(isupper(str[k]) && str[k] > 'E')  str[k] = tolower(str[k]);  else if(isspace(str[k]))  str[k] = '\*';  else if( ! isalnum(str[k]))  str[k] = '+';  else if(isdigit(str[k]) && str[k] != '9')  str[k] = str[k] + 1;  k++;  }  puts(str);  return 0;  } | **Output:**  **Bh\*479\*+Ac+rd**  Note: Full mark will also be given for the following solution:  **Bh\*479\*+AC+rd** |

|  |  |
| --- | --- |
| [5 points]  int k, m;  for(k = 1; k <= 5; k++){  printf("%d ", k);  for(m = 1; m <= k; m++){  printf("%d ", m + k);  }  printf("\n");  }  printf("%d %d", k, m); | **Output:**  **1 2**  **2 3 4**  **3 4 5 6**  **4 5 6 7 8**  **5 6 7 8 9 10**  **6 6** |

**Question # 3 [15 points: 13 + 2]**

1. The intersection of sets **A** and **B**, denoted by **A B**, is the set of elements that belong to both set **A** and to set **B**:

A B = { x such that x A and x



A one-dimensional array with no repeated elements can represent a set. Write a void function **setIntersection** that has six parameters: three one-dimensional integer array parameters **A**, **B, and C**, the size of array **A**, the size of array **B**, and an integer pointer parameter **numElements**. The function receives **A** and **B** as input parameters and it returns the set intersection **A B** in the array **C** and the number of elements of **C** in the parameter **numElements**.

Examples:

|  |  |  |  |
| --- | --- | --- | --- |
| Array A | Array B | Array C | numElements |
| {10, 20, 30 } | {5, 35, 45, 22} | (empty set) | 0 |
| {1, 2, 3 , 4 } | {3, 2, 4, 6} | {2, 3, 4} | 3 |
| {25, 8, 12, 70 } | {3, 70, 16, 25 } | {25, 70} | 2 |

**Note:**

* Your function must be general.
* Assume that both arrays **A** and **B** are filled with values and that each has no repeated element.
* Assume that size of array A is less or equal to the size of array B and that the size of array C is the same as the size of array A.
* Your function must not contain **scanf** and **printf** calls.
* DON’T WRITE THE **main** FUNCTION.

**(a) void setIntersection(int A[ ], int B[ ], int C[ ], int sizeA, int sizeB,**

**int\* numElements){**

**int k, m;**

**index = 0;**

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

**for(m = 0; m < sizeB; m++){**

**if(A[k] == B[m]){**

**C[index] = A[k];**

**index++;**

**break;**

**}**

**}**

**}**

**\*numElements = index;**

**}**

1. Assuming that integer arrays **A** and **B** are initialized and the number of elements in these arrays, namely, **sizeA**, **sizeB** are also initialized, also an integer array **C** and an integer **numElements** are declared, write an appropriate call to the function **setIntersection** that you were required to write in 3(a).

**Note:** You are only required to write the function call; leave the rest of the page blank.

**setIntersection(A, B, C, sizeA, sizeB, &numElements);**

**Question # 4 [15 points]**

Write a void function **getDistinctRowElements** that has the following input parameters:

* A full 2D-array **x** of type **double**.
* An integer variable **rowIndex**.

The function also has the following output parameters:

* An integer pointer that is used to return the count of the distinct elements in the row of **x** with index **rowIndex**. If **rowIndex** is not a valid row index for the array **x**, the function must return -1 in the output parameter.
* A 1-D array **y** of type double whose size is equal to the number of columns in array **x**. This array is used to return the distinct elements of the row of array **x** with index **rowIndex**. If **rowIndex** is not a valid row index for the array **x**, array **y** is left un-initialized.

**Example:**

If array **x** is:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 1.0 | 5.0 | 9.0 | 13.0 | 11.0 | 13.0 |
| 2.0 | 6.0 | 10.0 | 23.0 | 22.0 | 80.0 |
| 3.0 | 45.0 | 3.0 | 7.0 | 28.0 | 7.0 |
| 4.0 | 8.0 | 4.0 | 43.0 | 4.0 | 4.0 |
|  |  |  |  |  |  |

and rowIndex is **2**, then the returned array **y** is:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 3.0 | 45.0 | 7.0 | 28.0 |  |  |

And the returned count of the distinct elements is **4**

**Note:**

* Assume constants **NUMROWS and NUMCOLS** that store the number of rows and columns in array **x** are defined before all functions.
* The array **y** is passed as an empty un-initialized array.
* Your function must be general, it must not contain **scanf** and **printf** calls, and it must use appropriate **nested-loop**.
* DON’T WRITE THE **main** FUNCTION.

**void geDistinctRowElements(double x[ ][NUMCOLS], double y[ ], int rowIndex,**

**int\* numDistinctElements){**

**if(rowIndex < 0 || rowIndex > NUMROWS - 1){**

**\*numDistinctElements = -1;**

**return;**

**}**

**int count = 1;**

**int distinct, k, m;**

**y[0] = x[rowIndex][0];**

**for(k = 1; k < NUMCOLS; k++){**

**distinct = 1;**

**for(m = 0; m <= count; m++){**

**if(x[rowIndex][k] == y[m]){**

**distinct = 0;**

**break;**

**}**

**}**

**if(distinct){**

**y[count] = x[rowIndex][k];**

**count++;**

**}**

**}**

**\*numDistinctElements = count;**

**return;**

**}**

**Question # 5 [20 points]**

A textfile **input.txt** contains English words, unsigned integers, and signed integers. Each word or integer is separated from the next word or integer by one or more white space characters and/or punctuation characters. A sample input file is:

|  |
| --- |
| Is  the following:  3 25, 72, 651, 5000  a sequence of unsigned integers?  +32 and -67 are signed numbers, and  Jeddah, I assume, is 1245 kilometers from  Dhahran!  The time was 12:30:42 hours. |

Write a complete C program that reads the **input.txt** file and creates an output file **output.txt** in which:

* All the punctuation characters are removed.
* Each word or number is separated from the next word or number by a single blank.
* Each unsigned integer is reversed by using a function **reverseNumber**, that you must write.

The program also prints on the screen, the number of unsigned integers in the input file

Sample output.txt

|  |
| --- |
| Is  the following  3 52 27 156 0005  a sequence of unsigned integers  +32 and -67 are signed numbers and  Jeddah I assume is 5421 kilometers from  Dhahran  The time was 21 03 24 hours |

And the output on the screen is:

The count of unsigned integers is 9

**Hint:** White space characters are: blank, tab, and newline characters.

Punctuation characters are: **, ; . : ?** and **!**

**Note:**

* Your program must be general, it must work for an input file of any number of lines.
* Assume that the maximum line size in the text-file is 256.
* Your program must display an appropriate error message if the input file cannot be opened.

**#include <stdio.h>**

**#include <stdlib.h>**

**#include <string.h>**

**#include <ctype.h>**

**void reverseNumber(char word[]);**

**int main(void){**

**FILE \*infile, \*outfile;**

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

**if(infile == NULL){**

**printf("Error in opening input.txt");**

**exit(1);**

**}**

**outfile = fopen("output.txt", "w");**

**char line[257], delimeters[] = " .,;:!?\n\t";**

**char \*word;**

**int count = 0, k;**

**while( fgets(line, 257, infile) != NULL){**

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

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

**if(isdigit(word[0])){**

**reverseNumber(word);**

**count++;**

**}**

**fprintf(outfile, "%s ", word);**

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

**}**

**fprintf(outfile, "\n");**

**}**

**fclose(infile);**

**fclose(outfile);**

**printf("count of unsigned integers = %d\n", count);**

**return 0;**

**}**

**void reverseNumber(char word[]){**

**int size = strlen(word), k;**

**char temp;**

**for(k = 0; k < size/2; k++){**

**temp = word[k];**

**word[k] = word[size - 1 - k];**

**word[size - 1 - k] = temp;**

**}**

**}**

**// SEE ALTERNATE FUNCTION KEY ON THE NEXT PAGE:**

**/\* Alternate solution: Reversing using a temp array**

**void reverseNumber(char word[]){**

**int size = strlen(word), k;**

**char temp[size];**

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

**temp[size - 1 - k] = word[k];**

**}**

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

**word[k] = temp[k];**

**}**

**}**

**\*/**