KING FAHD UNIVERSITY OF PETROLEUM AND MINERALS

Information and Computer Science Department

2013 Fall Semester (Term 131)

ICS103 Computer Programming in C (2-3-3)

**Final Exam**

Jan5, 2014

120 Minutes

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Exam Code | **001** | | | | | | | | | | |
| Student Name |  | | | | | | | | | | |
| KFUPM ID |  |  | |  |  |  |  |  | |  |  |
| Class Section | ABDULLA AL-SUKAIRY | | □ 07 (UT 11am) | | |  | | |  | | |
| ASHARF AL\_FAGIH | | □ 11 (MW 07am) | | | □ 16 (MW 08am) | | | □ 19 (MW 09am) | | |
| AKRAM AHMAD | | □ 15 (MW 08am) | | | □ 18 (MW 09am) | | | □ 21 (MW 11am) | | |
| AHMAD IRFAN | | □ 13 (MW 07am) | | |  | | |  | | |
| AHMED AL-MULHEM | | □ 24 (MW 1:10pm) | | |  | | |  | | |
| EMAD RAMADAN | | □ 1 (UT 07am) | | | □ 5 (UT 08am) | | |  | | |
| ESAM MLAIH | | □ 2 (UT 07am) | | | □ 6 (UT 08am) | | | □ 10 (UT 1:10pm) | | |
| MASUD HASAN | | □ 17 (MW 09am) | | | □ 22 (MW 11am) | | | □ 25 (MW 1:10pm) | | |
| MOHAMED BALAH | | □ 3 (UT 07am) | | | □ 12 (MW 07am) | | |  | | |
| RAFI UL HASAN | | □ 8 (UT 11am) | | | □ 9 (UT 1:10pm) | | |  | | |
| ZAHID AYAR | | □ 20 (MW 11am) | | | □ 23 (MW 1:10pm) | | |  | | |
|  | |  | | |  | | |  | | |

**IMPORTANT NOTES**

* Fill-in your information on the answer sheet.
* **Answer all forty (40) questions.**
* **Mark your answers on the answer sheet.**
* **The answer sheet is the only one that will be graded.**
* Do NOT start the exam until you are instructed to do so.
* This is a closed material exam. So, remove any relevant material.
* Calculators are NOT allowed. If you have one, put it on the floor.
* Mobile phones are NOT allowed. If you have one, switch it off NOW.
* All questions carry equal weight

|  |  |  |
| --- | --- | --- |
|  | What is the type of function foo for which the prototype is shown below: | |
|  | void foo(int \*x, int \*y, int a, int b); | |
|  | A. | void function without arguments |
|  | B. | void function with input arguments |
|  | C. | Function with input arguments, single value returned |
|  | D. | Function with input arguments, multiple values returned |

|  |  |  |
| --- | --- | --- |
|  | What is a pointer variable? | |
|  | A. | A variable that stores the address of a normal variable |
|  | B. | A variable that can store a double value |
|  | C. | A variable that can have int or double or char value |
|  | D. | A variable used to declare a constant value |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | Given the following steps: | | | | | | | |
|  | Step 1: take input from the user and assign it to corresponding input variables  Step 2: print the results  Step 3: declare input/output variables  Step 4: perform calculation to obtain results  What will be the logical order of executing these steps in a C program? | | | | | | | |
|  | A. | Step 1  Step 2  Step 3  Step 4 | B. | Step 3  Step 1  Step 4  Step 2 | C. | Step 2  Step 4  Step 1  Step 3 | D. | Step 3  Step 4  Step 1  Step 2 |

|  |  |  |
| --- | --- | --- |
|  | What is the output after executing the following code? | |
|  | void f1(int \*x, int a, int b);  int main (void){  int x = 10, i, a = 0, b = 1;  for (i = 0; i < 2; i++)  f1(&x, a, b);  printf("%d %d", x, b);  return 0;  }  void f1(int \*x, int a, int b){  \*x = a + b;  b++;  } | |
|  | A. | 10 2 |
|  | B. | 1 1 |
|  | C. | 10 1 |
|  | D. | 1 2 |

|  |  |  |  |
| --- | --- | --- | --- |
| **The next two questions are based on the following code fragment:**  The code shown below is an incomplete program. It calls the get\_val function that reads a value from the user then the main function prints it. Select from the two questions shown below the proper statements to have a working program:  #include<stdio.h>  **\_\_\_\_\_Statement 1: Prototype\_\_\_\_\_\_\_;**  int main (void){  int inp;  **\_\_\_\_Statement 3: Function Call\_\_\_\_\_\_\_\_\_\_\_\_\_**  printf("you entered %d\n", inp);  return 0;  }  **\_\_\_\_\_\_\_ Statement 1: Function Header\_\_\_\_\_\_**{  **\_\_\_\_\_Statement 2: Function Body\_\_\_\_\_**  } | | | |
|  | What should be Statement 1 and 2 | | |
| A. | Statement 1: int get\_val(int \*x)  Statement 2: scanf("%d",&x); | B. | Statement 1: void get\_val(int \*x)  Statement 2: scanf("%d", \*x); |
| C. | Statement 1: void get\_val(int \*x)  Statement 2: scanf("%d", x); | D. | Statement 1: void get\_val(int \*x)  Statement 2: return scanf("%d", x); |
|  | What should be Statement 3 | | |
| A. | inp = get\_val(); | B. | get\_val(\*inp); |
| C. | get\_val(&inp); | D. | get\_val(inp); |

|  |  |  |
| --- | --- | --- |
| **The next two questions are based on the following code fragment:**  int x = 100, y = 10;  int \*p1, \*p2;  p1 = &x;  p2 = &y;  x = \*p1 + \*p2;  y = x - \*p2;  \*p1 = \*p1 - \*p2; | | |
|  | What is the final value of x? | |
|  | A. | 11 |
|  | B. | 10 |
|  | C. | 12 |
|  | D. | 9 |
|  | What is the final value of y? | |
|  | A. | 100 |
|  | B. | 110 |
|  | C. | 220 |
|  | D. | 230 |

|  |  |  |
| --- | --- | --- |
|  | Consider the **selection sort** function. Select the order of the elements of the array after each pass to sort the following array: {3,1,-1,0,2} | |
|  | A. | {-1,1,0,2,3} {-1,1,0,2,3} {-1,0,1,3,2} {-1,0,1,2,3} |
|  | B. | {-1,3,1,0,2} {-1,0,3,1,2} {-1,2,0,1,3} {-1,0,1,2,3} |
|  | C. | {-1,1,3,0,2} {-1,0,3,1,2} {-1,0,1,3,2} {-1,0,1,2,3} |
|  | D. | {-1,1,3,0,2} {-1,1,3,0,2} {1,-1,0,3,2} {-1,0,1,2,3} |

|  |  |  |
| --- | --- | --- |
|  | What is the output after executing the following code? | |
|  | void good(int x, int y);  int main (void){  double x = 10.0, y = 1.1239;  good(x,y);  return 0;  }  void good(int x, int y){  printf("%.2f", (double) x \* y);  } | |
|  | A. | 11.23 |
|  | B. | 11.24 |
|  | C. | 10.00 |
|  | D. | 11.00 |

|  |  |  |
| --- | --- | --- |
|  | What is the output after executing the following code? | |
|  | int f2(int a, int i);  int main (void){  int x = 3, a = 2;  x = f2(a, x);  printf("%d", x);  return 0;  }  int f2( int a, int b){  return a \* b;  } | |
|  | A. | 5 |
|  | B. | 3 |
|  | C. | 6 |
|  | D. | 4 |

|  |  |  |
| --- | --- | --- |
|  | Consider the following array: | |
|  | int x[7]={2, -1, 5, 3, 0, 4, 8};  Consider a call to linear search function covered in class with array x and target value of 6. How many times is the target value 6 compared inside the function? | |
|  | A. | 5 times |
|  | B. | 7 times |
|  | C. | 8 times |
|  | D. | 6 times |

|  |  |  |
| --- | --- | --- |
|  | The proper prototype for a function that adds two arrays A and B of size **(**n**)** and put the result back in array A is: | |
|  | A. | void addArray(int A[], int B[],int n); |
|  | B. | int A[] addArray(int A[], int B[], int n); |
|  | C. | void addArray(int A[], int B[], int \*A[], int n); |
|  | D. | void addArray(int A[n], int B[n], int n); |

|  |  |  |
| --- | --- | --- |
|  | What is the output after executing the code fragment shown below? | |
|  | int A[6] = {8,11,12,19,24,33};  int i, sum=0;  for (i=0; i<6; i++){  if (A[i]%2 ==0 && A[i]%3==0)  continue;  printf("%d ", A[i]);  sum += A[i];  if (sum>20)  break;  } | |
|  | A. | 8 11 12 19 24 33 |
|  | B. | 8 11 19 33 |
|  | C. | 8 11 19 |
|  | D. | 8 11 |
|  |  |  |

|  |  |  |
| --- | --- | --- |
|  | Consider the **bubble sort** function. Select the correct order of the elements of the array after each pass to sort the following array: {4,3,1,5,-2} | |
|  |  | |
|  | A. | {-2,4,3,1,5} {-2,1,4,3,5} {-2,1,3,4,5} {-2,1,3,4,5} |
|  | B. | {-2,4,3,1,5} {-2,1,3,5,4} {-2,1,3,5,4} {-2,1,3,4,5} |
|  | C. | {3,1,4,-2,5} {1,3,4,-2,5} {-2,1,4,3,5} {-2,1,3,4,5} |
|  | D. | {3,1,4,-2,5} {1,3,-2,4,5} {1,-2,3,4,5} {-2,1,3,4,5} |

|  |  |  |
| --- | --- | --- |
|  | What will be the output of the following code fragment: | |
|  | int x[3][3]={{21,12,43},{24,55,76},{64,90,45}}, i, j;  for(i=2; i>=0; i--)  for(j=0; j<i; j++)  printf("%d ",x[i][j]); | |
|  | A. | 12 55 90 |
|  | B. | 64 90 24 |
|  | C. | 21 12 24 |
|  | D. | 43 12 64 |

|  |  |  |
| --- | --- | --- |
| **The next five questions are based on the following incomplete code**  The code shown below is for the **main** and **modify** functions. **modify** is a logical function that receives an integer array named x of size m and it swaps each element in the array with another element from the same array having a random index. The random index will be generated by calling rand() function used in the lab. The **modify** function returns 0 if the array is not modified otherwise it returns 1.  #include<stdio.h>  #include<stdlib.h>  **\_\_\_\_\_Statement 1: Prototype\_\_\_\_\_\_;**  int main() {  int x[8]={1,2,3,4,5,6,7,8};  if(**\_Statement 2: Function Call\_**)  printf("array modified");  else  printf("array not modified");  return 0;  }  //below is the function definition of modify  **\_\_\_\_\_\_\_ Statement 1: Function Header\_\_\_\_\_\_**{  int rand\_index, temp, i;  /\* array x will not be modified if its size is 1\*/  **\_\_\_\_\_Statement 3: Test Array Size\_\_\_\_**  for(i = 0; i< m; i++){  /\*Swap element at index i with element at random index \*/  **\_\_\_\_\_Statement 4: generate random index\_\_\_\_\_**  temp = x[rand\_index];  **\_\_\_\_\_Statement 5: swap\_\_\_\_\_**  x[i] = temp;  }  return 1;  } | | |
|  | Statement 1 should be: | |
|  | A. | int modify(int x[],int m) |
|  | B. | void modify (int x[],int m) |
|  | C. | void modify (int x,int m) |
|  | D. | int modify (int x[m]) |
|  | Statement 2 should be: | |
|  | A. | modify(x,8) |
|  | B. | void modify(x[], 8) |
|  | C. | modify (x[], 8) |
|  | D. | int modify (x,8) |
|  | Statement 3 should be: | |
|  | A. | if (m == 1) return 1; |
|  | B. | if (m = 1) return 0; |
|  | C. | if (m == 1) return 0; |
|  | D. | if (m = 1) return 1; |
|  | Statement 4 should be: | |
|  | A. | rand\_index = rand()%(m+1); |
|  | B. | rand\_index = rand() % m-1; |
|  | C. | rand\_index = rand(); |
|  | D. | rand\_index = rand() % m; |
|  | Statement 5 should be: | |
|  | A. | x[rand\_index] = temp; |
|  | B. | x[rand\_index] = x[i]; |
|  | C. | x[i]= x[rand\_index]; |
|  | D. | temp=x[i]; |

|  |  |  |
| --- | --- | --- |
|  | Consider the following array: | |
|  | int m[3][3] ={{44,12,1}, {7}, {56,4}};  What is the value of m[1][1]? | |
|  | A. | 12 |
|  | B. | 44 |
|  | C. | 4 |
|  | D. | 0 |

|  |  |  |
| --- | --- | --- |
|  | Consider the following incomplete code fragment: | |
|  | int x[3]={2,5,9};  **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_//** call the function fun  printf("%d %d %d\n", x[0],x[1],x[2]);  The function fun is defined as follows:  void fun (int \*a, int b, int \*c) {  \*a = \*a + 2;  b = b - 1;  \*c = \*c + 1;  }  Which of the following is the correct call for the function fun to generate: 2 6 11 as the output? | |
|  | A. | fun(&x[1], x[0], &x[2]); |
|  | B. | fun(&x[2], x[0], &x[1]); |
|  | C. | fun(&x[2], x[1], &x[0]); |
|  | D. | fun(x, x[0], x); |

|  |  |  |
| --- | --- | --- |
|  | Assume the following declaration and missing function call | |
|  | int x[5][7];  **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_// call to function** fun  The function fun has the following prototype:  void fun (int a[][7], int rows, int cols);  Which one of the following is the correct call to the function fun? | |
|  | A. | fun (x[][7], 5, 7); |
|  | B. | void fun (x[5][7], 5,7); |
|  | C. | fun (x,5,7); |
|  | D. | fun ( &x,5,7); |

|  |  |  |
| --- | --- | --- |
| **The next two questions are based on the following code fragment**  char line[20],x[4][7]={"I7S103","4m8","Gr6at!","KFUPM"};  int i, j, p=0, k=0;  for(i=0; i< 3 ;i++)  for(j=0; j < 3; j++)  if(isalpha(x[j][i]))  line[k++] = x[j][i];  else  p++;  line[k]='\0';  puts(line); | | |
|  | What is the output of the above code fragment? | |
|  | A. | ISmGr |
|  | B. | I4G7mrS86 |
|  | C. | 7486 |
|  | D. | IGmrS |
|  | What will be the final value of p after executing the code? | |
|  | A. | 4 |
|  | B. | 5 |
|  | C. | 3 |
|  | D. | 6 |

|  |  |  |
| --- | --- | --- |
| **The next two questions are based on the following code**  int x[4][4]={{2,6,-7,1},{7,-4,3,5},{6,8,-9,5},{3,6,9,-1}};  int max,i;  **\_\_\_\_\_Statement 1\_\_\_\_\_**  for(i=1; i<=3; i++)  **\_\_\_\_\_Statement 2\_\_\_\_\_**  max = x[i][i];  The objective of the above code is to find the maximum value of the diagonal of array x and save it in variable max. What should be the missing 2 statements? | | |
|  | Statement 1 should be: | |
|  | A. | max = x[0][0]; |
|  | B. | max = x[0][4]; |
|  | C. | max = x[2][0]; |
|  | D. | max = x[0][3]; |
|  | Statement 2 should be: | |
|  | A. | if(x[i-1][i-1] < max) |
|  | B. | if(max > x[i+1][i+1]) |
|  | C. | if(x[i][i] > max) |
|  | D. | if(x[4][4] < max) |

|  |  |  |
| --- | --- | --- |
|  | What will be the output of the following statement? | |
|  | strcmp("Az","AbdulRahman"); | |
|  | A. | 1 |
|  | B. | 0 |
|  | C. | -1 |
|  | D. | None of the above |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | Which code fragment will assign the following values to array **x**? | | | |  |  |  |  | | --- | --- | --- | --- | | 0 | 9 | 9 | 9 | | 7 | 0 | 9 | 9 | | 7 | 7 | 0 | 9 | |
|  |  | | |
| A. | int i,j, x[3][4]={0};  for (i=0; i<3; i++) {  for (j=0; j<4; j++) {  if(i> j)  x[i][j]=7;  else  x[i][j]=9;  }} | B. | int i,j, x[3][4];  for (i=0; i<3; i++) {  for (j=0; j<4; j++) {  if(i> j)  x[i][j]=7;  else if (i< j)  x[i][j]=9;  }} | |
| C. | int i,j, x[3][4]={0};  for (i=0; i<3; i++) {  for (j=0; j<4; j++) {  if(i> j)  x[i][j]=7;  else if (i< j)  x[i][j]=9;  }} | D. | int i,j, x[3][4]={0};  for (i=0; i<3; i++) {  for (j=0; j<4; j++) {  if(i>= j)  x[i][j]=7;  else  x[i][j]=9;  }} | |

|  |  |  |
| --- | --- | --- |
|  | What is the output of the following code fragment? | |
|  | char m[][15] = {"Welcome#to#","ICS#103","Students"};  int i,j=0;  for(i=0; i<3; i++){  printf("%c %d ", m[i][j], strlen(m[i]));  j++;  } | |
|  | A. | W 11 C 7 u 8 |
|  | B. | W 10 S 7 t 8 |
|  | C. | W 11 I 7 S 8 |
|  | D. | w 11 C 6 d 7 |

|  |  |  |
| --- | --- | --- |
|  | Suppose an array is initialized with the following values {1, 2, 3, 4, 5, 6, 4}. When searching this array for target value 4, the call to the linear search function covered in class will return: | |
|  | A. | 6 |
|  | B. | 7 |
|  | C. | 4 |
|  | D. | 3 |

|  |  |  |
| --- | --- | --- |
| Fill in the gaps such that the following code prints this string: **DDKKTT**  int main() {  char s[]="ADGKMT";  int i;  for (i=0;s[i]!='\0';i++) {  if(i%2==0)  **\_\_\_\_\_Statement 1\_\_\_\_\_**  }  puts(s);  return 0;  } | | |
|  | Statement 1 should be: | |
|  | A. | s[i]=s[i+1]; |
|  | B. | s[i+1] = s[i]; |
|  | C. | strcat(s,"DDKKTT"); |
|  | D. | strcpy(s[i],s[i+1]); |

|  |  |  |
| --- | --- | --- |
|  | What is the output of the following code fragment? (abcdefghijklmnopqrstuvwxyz) | |
|  | char text[3][10]={"abc","ns","mk"};  int i;  for(i=2;i>0;i--) {  if(myFun(text[i],text[i-1])>0)  strcat(text[i-1],text[i]);  else if (myFun (text[i],text[i-1])<0)  strcat(text[i],text[i-1]);  else  strcpy(text[i],text[i-1]);  puts(text[i]);  }  The definition of myFun is given below:  int myFun (char s1[], char s2[]) {  return strlen(s1)-strlen(s2);  } | |
|  | A. | mkns  abc |
|  | B. | ns  nsabc |
|  | C. | mk  nsabc |
|  | D. | ns  abcns |

|  |  |  |
| --- | --- | --- |
| **The next two questions are based on the following code fragment**  Fill in the blanks of the following code. The code reads a single character from the user, then it will display one of the 3 messages depending on the character itself:  - It is an alphabetic character  - It is a digit  - It is a special character  Note: A special character is a character that is neither an alphabet nor a digit.  int main() {  char c;  printf("Enter any character>");  scanf("%c",&c);  if(**\_\_\_\_\_Condition 1\_\_\_\_\_**)  printf("It is an alphabet character");  else if(**\_\_\_\_\_Condition 2\_\_\_\_\_**)  printf("It is a digit");  else  printf("It is a special character");  return 0;  } | | |
|  | Condition 1 should be: | |
|  | A. | isalnum(c) |
|  | B. | c=isalpha |
|  | C. | c==isalpha() |
|  | D. | isalpha(c) |
|  | Condition 2 should be: | |
|  | A. | isdigit(c) |
|  | B. | c==isdigit() |
|  | C. | isdigit==c |
|  | D. | none of the above |

|  |  |  |
| --- | --- | --- |
|  | What is the output of the following code fragment? | |
|  | char s[12] ="easy#exam#";  int i;  for (i=0; s[i]!='\0'; i++)  if (s[i]=='#')  printf("%d ", strlen(s)-i); | |
|  | A. | 6 1 |
|  | B. | 6 3 |
|  | C. | 8 4 |
|  | D. | 5 2 |

|  |  |  |
| --- | --- | --- |
| **The next two questions are based on the following code**  char \*token, result[80] = "", delim[] = "&#";  char str[] ="ICS&103#IS&A#GREAT&COURSE#";  token = strtok(str, delim);  while(token != NULL){  if(strlen(token)>3 && !isdigit(token[0]))  printf("%s",token);  strcat(result, token);  token = strtok(NULL, delim);  } | | |
|  | What is the output after executing the following code fragment? | |
|  | A. | ICS COURSE |
|  | B. | ICS GREAT COURSE |
|  | C. | ICS 103 GREAT COURSE |
|  | D. | GREATCOURSE |
|  | What will be printed if the following statement is added after the while loop?  puts(result); | |
|  | A. | ICS&103#IS&A#GREAT&COURSE# |
|  | B. | GREATCOURSE |
|  | C. | ICS103ISAGREATCOURSE |
|  | D. | ICSISAGREATCOURSE |

|  |  |  |
| --- | --- | --- |
|  | What will be the output of the following code fragment? | |
|  | int z[2] = {7, 9};  ICS(z, z[1]);  printf("%d %d", z[0], z[1]);  where the function ICS is defined as:  void ICS(int ar[],int r){  ar[0] = 2;  r= 4;  } | |
|  | A. | 2 4 |
|  | B. | 7 4 |
|  | C. | 7 9 |
|  | D. | 2 9 |