# King Saud University

**College of Computer and Information Sciences**
**Computer Science Department**

| | | |
|---|---|---|
| | **Course Code:** | **CSC 215** |
| | **Course Title:** | **Procedural Programming** |
| | **Semester:** | **Fall 2016 (Semester 1 of 2016-2017)** |
| | **Final Exam** | |
| | **Duration: 120 minutes** | |
| **Student Name:** | | |
| **Student ID:** | | |
| **Student Section No.** | | |

| | **Computer Science B.Sc. Program:**<br>NCAAA: Intended Learning Outcomes (ILO) Student Outcomes<br>ABET: Program Learning Outcomes (PLO) Student outcomes | **Question No. Relevant is Hyperlinked** | **Covering %** |
|---|---|---|---|
| NCAAA | **1. Knowledge (NCAAA)**<br>Suggested verbs (list, name, record, define, label, outline, state, describe, recall, memorize, reproduce, recognize, record, tell, write) | Exercises:<br>Q1: 1 through 6<br>Q2: 2, 6, 8, 10 | |
| ABET | (i) Use current techniques, skills, and tools necessary for computing practices;<br>*The students learn how to use Integrated Development Environment to compile and run C programs. Students also learn the differences between procedural and object oriented languages* | Exercises:<br>Q1: 1 through 6<br>Q2: 2, 6, 8, 10 | |
| NCAAA | **2. Cognitive Skills (NCAAA)**<br>Suggested verbs (estimate, explain, summarize, write, compare, contrast, diagram, subdivide, differentiate, criticize, calculate, analyze, compose, develop, create, prepare, reconstruct, reorganize, summarize, explain, predict, justify, rate, evaluate, plan, design, measure, judge, justify, interpret, appraise) | Exercises:<br>Q1: 7 through 10<br>Q2: 1,3,4,5,7,9<br>Q3 and Q4 | |
| ABET | b. An ability to analyze a problem, and identify and define the computing requirements appropriate to its solution;<br>*Students learn how to read and analyze a problem, determine its input and output, and allocated the required storage statically or dynamically.* | Exercise:<br>Q1: 9<br>Q2: 1, 8<br>Q3: D1<br>Q4: 3 | |
| | c. An ability to design, implement and evaluate a computer-based system, process, component or program to meet desired goals.<br>*Students learn how to analyze a procedural c program, evaluate c expressions and interpret algorithmic steps from natural language into c programs.* | Exercises:<br>Q1: 7, 8 and 10<br>Q2: 2 - 7, 9, 10<br>Q3, A, B, C, D2<br>Q4: 1, 2, 4 - 6 | |

**Question1:** Write True/ False **( Total 10 points)**

| | Statement | True or False |
|---|---|---|
| 1 | If an automatic variable and a global one have identical names, references to the name within the scope where both are valid refer to the global variable | |
| 2 | All unary operators have the same precedence, with associativity from right to left. | |
| 3 | The time complexity of sorting any `n`-element array using quicksort algorithm is O(nlogn). | |
| 4 | The library function `free` releases the memory pointed to by its pointer parameter and sets the parameter to NULL. | |
| 5 | Bubble sort and selection sort algorithms have the same number of swap operations in each iteration, but bubble sort perform more comparisons. | |
| 6 | Size of a union is equal or greater than the sum of the individual sizes of its members. | |
| 7 | If the integer `s` is 10, then the expression `s = (s>=10) && (s=25)` evaluates to 1 | |
| 8 | The two statements: `scanf("%c", &letter);` and `letter = getchar();` do exactly the same thing. | |
| 9 | The statements: `int* x; *x=5;` are the correct way to store the integer value 5 in a memory location addressed by x. | |
| 10 | To test whether x is an ascii code of an uppercase letter, one can use: `if ('A' <= x && x <= 'Z')` | |

**Question 2:** Select the correct answer **(Total 10 points)**

1. How many bytes are required to store the string `"without \n doubt"` ?

   A. 13
   B. 14
   C. 15
   D. 16

---

2. The output of the **rand() % 50** is:

   A. random integer between 1 and 50
   B. random integer between 0 and 49
   C. random integer less than 50 including negative numbers
   D. none of the above.

3. What will be printed by the following program?

```
01   #include <stdio.h>
02   void f(int x){
03     x = x + 5 * x * x;
04   }
05   int main(){
06     int x = 5;
07     printf("%d", x);
08     return 0;
09   }
```

A. 5
B. 130
C. 150
D. 250

---

4. What will be printed by the following program?

```
#include <stdio.h>
int main(){
  int x=3, y=5;
  if (x > y)
    printf("A");
  if (x = 4)
    printf("%d\n", x+y);
  return 0;
}
```

A. `A`
B. `A8`
C. `A`
   `8`
D. `9`

---

5. Assume we included the required header files, which of the following is a wrong way to initialize the elements of integer array x to 0s?

A.
```
int x[10]={0};
```
B.
```
int x[10];
x = {0, 0, 0, 0, 0, 0, 0, 0, 0, 0};
```
C.
```
int x[10];
memset(x, 0, sizeof(x));
```
D. `none of the above.`

---

6. If function `foo` is defined as:

```
float foo(int m){
  return m + m/10.0 + m/100.0;
}
```

and the pointer fp is declared as:

```
float (*fp)(int);
```

then, the right way to assign foo to fp is:

A. `fp = foo;`
B. `fp = &foo;`
C. `fp = *foo;`
D. all of the above

---

7. What is the output of the following code fragment?

```
int a[] = {10, 20, 30 ,40, 50}, *c, *b=a;
printf("%i ", *(b++));
c = b;
printf("%i ", ++(*c));
```

A. `10 21`

B. `20 21`

C. `10 20`

D. `11 20`

---

8. The safe function to avoid the overflow condition when reading a string from the keyboard is:

A. `gets`

B. `fgets`

C. `scanf`

D. `fscanf`

---

9. How can you write `a[i][j][k][l]` in equivalent pointer expression?

A. `(((***(a+i)+j)+k)+l)`

B. `((**(*(a+i)+j)+k)+l)`

C. `(*(*(*(a+i)+j)+k)+l)`

D. `*(*(*(*(a+i)+j)+k)+l)`

---

10. When a break statement is encountered within a loop body,

A. the execution of the loop body is interrupted, and the program control transfers to the exit point of the loop.

B. all the remaining statements in the loop body are skipped and the loop continuation condition is evaluated next.

C. the program stops.

D. nothing happens.

**Question 3:** Answer the following questions based on its given code: **(Total 11 points)**

**A.** (2.5 points)

```
01 #include <stdio.h>
02 int main() {
03   int fibArr[] = {1, 2, 3, 5, 8, 13, 21, 34, 55, 89};
04   int primArr[] = {2, 3, 5, 7, 11, 13, 17, 19, 23, 29};
05   int commArr[], i, j;
06   for (i=0; i<10; ++i){
07     for (j=0; j<10; ++j)
08       if (fibArr[i] = primArr[j]){
09         commArr[j] = primArr[j];
10         ++n;
11       }
12     }
13   printf("Total number of common elements is %n\n", n);
14   return 0;
15 }
```

The program finds the number of common elements in two different integer arrays (`fibArr` and `primArr`) and stores them in another array `commArr`, then prints out how many common elements there are. There are five errors in the code. Identify them and fix them.

| Line# | Error description | Fix |
|---|---|---|
| | | |
| | | |
| | | |
| | | |
| | | |

**B.** (3.5 points)

```
for (i=0; i < 10; i++)
  x[i] = i + 1;
```

Suppose `x` is an array of integers, and `x[0]` is stored at address `0x4500`. What is the value of the expressions:

i. `x` ……

ii. `&x[0]` ……

iii. `*x` ……

iv. `x[1]` ……

v. `&x[1]` ……

vi. `sizeof(x)` ……

vii. `sizeof(x+3)` .....

**C.** (2.0 points)

```
01 #include <stdio.h>
02 int main(){
03   int a = 10, b = 9, c = 8;
04   printf("a > b: %d\n", a>b);
05   printf("a-c==b+c : %d\n", a-c==b+c);
06   printf("a+=b!=c: %d\n" , a+=b!=c);
07   return 0;
08 }
```

Write the output of the above C Program :

**D.** (3.0 points)

```
void fetch(int* arr, int count){
  if (count==1)
    printf("%i ", arr[count-1]);
  else fetch(arr+1, count-1);
}
```

1. if the function is called from `main` using the expression `fetch(arr, 5)`, how many copies of the variable `count` will be on the memory stack when the base case is reached?

   .....

2. Rewrite the function fetch in an iterative style, equivalent to the recursive given here.

**Question 4:** Consider the definitions: **(Total 9 points)**

```
typedef struct Node{ int data; struct Node* next; } Node;
typedef struct LinkedList{ Node* head; } linkedList;
int comp(const void* a, const void* b){
  return *(int*)a - *(int*)b;
}
```

The standard library offers a quick sort implementation through the function:

```
void qsort(void *base, size_t nmemb, size_t size,
           int (*compar)(const void *, const void *));
```

This function can sort arrays, but not linked-lists. So, we will write a function `print_sorted_ll` that takes a LinkedList parameter `ll` and prints the data stored in its nodes in ascending order, using the steps:

1. Write the header of the function print_sorted_ll

   .....

2. Count the nodes in the list ll in a variable named `count`

3. Dynamically allocate an array `arr` big enough to hold all the data stored in `ll`

   .....

4. Copy the data stored in `ll` to the array `arr`

5. Sort the array using the function `qsort`

6. Iterate over the sorted array and print the data

## Bonus Question (2 points)

The stack structure is utilized through the functions:

```
int is_empty(struct stack s);

int peak(struct stack s);

void push(struct stack* s, int data);

int pop(struct stack* s);
```

If stack S1 contains integer values in ascending order (i.e the smallest value resides at the top of the stack) and stack S2 contains integers in ascending order, write a function rev_merge that returns a stack structure that contains all the elements of S1 and S2 in descending order, using the above functions only, independently from the internal implementation of the stack.