CS 210 **Name SOLUTION**
Hour Exam 2 **October 18, 2016**

1. If an array is defined by the statement `int A[30];` , answer the following questions:

A) How many integers can be stored in the array? **30**

B) Write a statement to store the number 54 in the last element of the array.

```
A[29] = 54;
```

C) If the user write the statement below, what are the possible results? Explain.

```
A[34] = 22;
```

A[34] does not exist so this will likely cause a memory fault or possibly set another variable to the value of 22.

D) Write a short sequence of statements to exchange the values in element `A[12]` with the value in `A[11]`.

```
int tmp;
tmp = A[12];
A[12] = A[11];
A[11] = tmp;
```

2. C does not do bounds checking on array sizes. What does this mean and what are the implications for C programmers.

It means that when an array is defined with a size, the C-compiler does not check to see that all array operations stay within the size boundary. This may result in a program seeding access to a memory location which it does not own creating an error.

3. Show what is printed by the following program. Assume that the variable u is stored in memory at 7339360 and v is stored at 7339348.

```
#include<stdio.h>
int main()
  {int u = 3, v;
   int *ptrU, *ptrV;
   ptrU = &u;
   v = *ptrU;
   ptrV = &v;
   printf("u = %d v = %d\n", u, v);
   printf("%d, %d\n", ptrU, ptrV);
   printf("%d, %d\n", &u, &v);
   return 0;
  }
```

Printed Results

```
u = 3 v = 3
7339360, 7339348
7339360, 7339348
```

4. Mark each statement *A* to *D* below as true or false. The statement apply after the following code fragment runs.

```
int [] x = {9, 8, 7, 6};      //line 1
int [] y = {12, 15, 4, 3};    //line 2
x[2] = 4;                     //line 3
y[0] = x[3];                  //line 4
x[y[3]] = 12;                 //line 5
```

A) line 5 is illegal **FALSE**
B) x[1] = 8 **TRUE**
C) y[3] = 4 **FALSE**
D) y[0] = 9 **FALSE**

5. Show what is printed by the following program.

```
#include <stdio.h>
int MFun(int *, int);
int main()
 {int a = 5;
  int *aptr;
  aptr = &a;
  int b = 0, c = 0;
  c = MFun(aptr, b);
  printf("%d %d %d\n", a, b, c);
  return 0;
 }
//
int MFun(int *x, int y)
 {y = *x * *x;
  printf("%d %d\n", *x, y);
  return y;
 }
```

Printed results

5 25

5 0 25

6. Write a function which will return the average value of a one-dimensional int array passed as a parameter. Name your method FindAverage. You may pass the array size as a second parameter.

```
double FindAverage(int a[], int n)
  {int i, sum = 0;
   for(i=0;i<n;i++)
      sum += a[i];
   return (double)sum/n;
  }
```

7. The following statements creates an array and fills it with random numbers. Write a sequence to find and print the *array index* of the item which has the least absolute value in the array.

```
srand(23);
int a[203];
int i;
for(i=0;i<203;i++)
  a[i] = (rand() % 25) -12;
```

```
int min, minIndex;
minIndex = 0;
min = abs(a[minIndex]);
for(i=1;i<203;i++)
   {if(min > abs(a[i]))
       {min = abs(a[i]);
        minIndex = i;
       }
   }
printf("%d\n", minIndex);
```

8. The program below is a method which does a *select sort*. This method calls two other methods named *FindMin* and *Swap*. Answer the questions below about these three methods.

```
void SelectionSort(int d[], int n)
    {int i, j, minIndx;
     for(i=0;i<n-1;i++)
        {for(j=i;j<n;j++)
           {FindMin(d, n, i, &minIndx);
            Swap(&d[i], &d[minIndx]);
            }
        }
    }
void FindMin(int d[], int n, int start, int *minIndx)
   {int i;
    *minIndx = start;
    for(i=start;i<n;i++)
       {if(d[i] < d[*minIndx])
           *minIndx = i;
       }
   }
void Swap(int *a, int *b)
   {int tmp;
    tmp = *a;
    *a = *b;
    *b = tmp;
   }
```

A) What will be in the array x if we execute the following two statements?

```
int x[] = {2, 8, 10, 6, 4};
SelectionSort(x, 5);
```

x = {2, 4, 6, 8, 10};

B) Does the Sort program still work correctly if two entries of the array it is sorting have the same value? For example:

```
int y[] = {2, 8, 10, 8, 4};
SelectionSort(y, 5);
```

Explain why or why not?

Yes it does work correctly. If two values are the same they will appear adjacent to each other in the final sorted array.

C) Show how you could use the FindMin method to find and print the minimum of the array given by

```
int z[] = {2, 8, 10, 6, 4, 0, -6, 15};
```

```
int minIndx;
FindMin(z, 8, 0, &minIndx);
printf("%d\n", z[minIndx]);
```

D) What is in array x after the following sequence runs:

```
int x[] = {2, 8, 10, 6, 4};
Swap(*x[2], *x[0]);
```

x = {10, 8, 2, 6, 4);