

# ECE 264 Exam 1 Answer

**01:30-02:20AM, February 06, 2009**

## 1 Control Flow (1.5 points)

What is the output of this program?

```
#include <stdio.h>
int f1(int x, int * y)
{
    if (x > 0)
    {
        * y = -x;
        return (x + 1);
    }
    else
    {
        * y = x + 1;
    }
    return (x - 1);
}
int main(int argc, char* argv[])
{
    int v1 = 1;
    int v2 = 2;
    int v3 = 3;
    int v4 = 4;
    if (v1 > 0)
    {
        v2 = f1(v3, & v4);
        printf("A %d %d %d %d\n", v1, v2, v3, v4);
        if ((v2 > v3) || (f1 (v2, & v4) > 0))
        {
            printf("B %d %d %d %d\n", v1, v2, v3, v4);
        }
        else
        {
            printf("C %d %d %d %d\n", v1, v2, v3, v4);
        }
    }
    else
```

Name:

1

Seat:

```

{
    v2 = f1(v4, & v3);
    printf("D %d %d %d %d\n", v1, v2, v3, v4);
    if ((v1 > v2) && (f1 (v3, & v2) > 0))
    {
        printf("E %d %d %d %d\n", v1, v2, v3, v4);
    }
    else
    {
        printf("F %d %d %d %d\n", v1, v2, v3, v4);
    }
}
v1 = f1(v4, & v3);
printf("G %d %d %d %d\n", v1, v2, v3, v4);
return 0;
}

/* Hint: C uses "short-circuit evaluation" */

```

*Answer:*

A 1 4 3 -3  
B 1 4 3 -3  
G -4 4 -2 -3

## 2 Memory Allocation (0.6 points)

Allocate an integer array of 26 elements using `malloc`, assign 264 to each element, and release the memory.

*Answer:*

`int index;`

Name:

2

Seat:

```

int * intA = malloc(26 * sizeof(int));
for (index = 0; index < 26; index++)
{
    intA[index] = 264;
}
free(intA);

```

### 3 Modified Bubble Sort (3 points)

Assume a swap function for two integers is available. Modify bubble sort for an integer array of  $n$  elements so that

- The smallest element is the first element (index = 0).
- The second smallest element is the last element (index =  $n - 1$ ).
- The third smallest element is the second element.
- The fourth smallest element is the second last element.
- The  $i^{th}$  smallest element is the  $(\frac{i+1}{2})^{th}$  element (index =  $\frac{i-1}{2}$ ), when  $i$  is an odd number.
- The  $i^{th}$  smallest element is the  $(\frac{i}{2})^{th}$  last element (index =  $n - \frac{i}{2}$ ), when  $i$  is an even number.

One simple solution is to use bubble sort first and then reassign the elements based on the rules. You will receive **no point** if you do this. You have to modify bubble sort directly.

*Answer:*

```

/* bubblesort.c */
#include <time.h>
#include <stdio.h>
#include <stdlib.h>
void swap(int *p, int *q)
{
    int tmp;
    tmp = *p;
    *p = *q;
    *q = tmp;
}

void printArray(int a[], int n)
{
    int i, j;

```

```

for (i = 0; i < n; ++i)
{
    printf("%d ", a[i]);
}
printf("\n");
/* visualize */
for (i = 0; i < n; ++i)
{
    printf("%3d ", i);
    for (j = 0; j < a[i]; j++)
    {
        printf("*");
    }
    printf("\n");
}
printf("\n");
}

/* original */
void bubbleSort1(int a[], int n)
{
    int i, j;
    for (i = 0; i < n - 1; ++i)
    {
        for (j = i + 1; j < n; j++)
        {
            if (a[i] > a[j])
            {
                swap(&a[i], &a[j]);
            }
        }
    }
}

/*
i = 0, smallest between 0 and n - 1, assigned to 0
i = 1, smallest between 1 and n - 1, assigned to n - 1
i = 2, smallest between 1 and n - 2, assigned to 1
i = 3, smallest between 2 and n - 2, assigned to n - 2
i = 4, smallest between 2 and n - 3, assigned to 2
i = 5, smallest between 3 and n - 3, assigned to n - 3
i = 6, smallest between 3 and n - 4, assigned to 3
...
i is even, smallest between i/2 and n - i/2 - 1, assigned to i/2
i is odd, smallest between (i + 1)/2 and n - (i + 1)/2,

```

```

                                assigned to n - (i + 1) / 2
*/



/* answer for the exam */
void bubbleSort2(int a[], int n)
{
    int i, j, head, tail;
    for (i = 0; i < n - 1; i++)
    {
        if ((i % 2) == 0)
        {
            head = i / 2;
            tail = n - 1 - i / 2;
        }
        else
        {
            head = (i + 1) / 2;
            tail = n - (i + 1) / 2;
        }
        for (j = head; j <= tail; j++)
        {
            if ((i % 2) == 0)
            {
                if (a[head] > a[j])
                {
                    swap(&a[head], &a[j]);
                }
            }
            else
            {
                if (a[tail] > a[j])
                {
                    swap(&a[tail], &a[j]);
                }
            }
        }
    }
}

/* main function not necessary */
int main(int argc, char * argv[])
{
    int * intArray;
    int numElem;
    int elemCnt;

```

```

if (argc < 2)
{
    printf("need one number for the size\n");
    return -1;
}
numElem = strtol(argv[1], NULL, 10);
if (numElem <= 0)
{
    printf("invalid size\n");
    return -1;
}
intArray = malloc(numElem * sizeof(int));
if (intArray == NULL)
{
    printf("memory allocation fail\n");
    return -1;
}
srand (time(NULL));
for (elemCnt = 0; elemCnt < numElem; elemCnt++)
{
    intArray[elemCnt] = 1 + rand() % 100;
}
bubbleSort1(intArray, numElem);
printArray(intArray, numElem);
bubbleSort2(intArray, numElem);
printArray(intArray, numElem);
return 0;
}

/*
example (n is even)

2 8 13 25 29 29 46 57 61 68
0 **
1 *****
2 *********
3 *********
4 *********
5 *********
6 *********
7 *********
8 *********
9 *********

```

2 13 29 46 61 68 57 29 25 8

```

0 **
1 *****
2 *****
3 *****
4 *****
5 *****
6 *****
7 *****
8 *****
9 *****

example (n is odd)

2 3 5 24 35 49 52 55 61
0 **
1 ***
2 ****
3 *****
4 *****
5 *****
6 *****
7 *****
8 *****

2 5 35 52 61 55 49 24 3
0 **
1 ****
2 *****
3 *****
4 *****
5 *****
6 *****
7 *****
8 ***

*/

```

## 4 UNIX Commands (1 point)

For each command on the left column, find the best one match on the right column.

Name:

7

Seat:

command	option
man gcc	A. change to the directory called music B. print all files created today C. execute Makefile D. commit the changes to the repository E. show the manual of gcc F. create a directory called music G. list files and directories H. check whether program music leaks memory I. print the number of lines in file music K. play the music in the CDROM M. list hidden files P. remove two files called xyz and stu Q. compare the two files xyz and stu line by line S. use gcc to compile a program called man.c U. check out a directory called gcc from the repository X. list the titles of the songs in the CDROM Z. rename file xyz (source) to stu (destination) 1. copy file stu (source) to file xyz (destination) 2. replace xyz by stu 3. call gcc in Makefile 4. print the path of the current directory 5. copy file xyz (source) to file stu (destination)
ls	
cd music	
pwd	
rm xyz stu	

Answer: man- E, ls- G, cd- A, pwd- 4, rm- P. 0.2 point for each correct answer.

## 5 Outcome 1, File (3.9 points)

Write a function that takes two input arguments:

- Both arguments have type `char *`.
- The first argument is the name of an *input* file.
- The second argument is the name of an *output* file.

If opening either file fails, the function returns -1. Otherwise, the function reads the input file line by line. For each line counts how many times "ECE 264" occurs, and writes the number to the output file. Each line may contain zero, one, or several "ECE 264". If "ECE 264" is divided into two or more lines, it is **not** counted. Close both files before the function returns. If the function completes successfully, it returns 0.

You can use the `strstr` and `fgets` functions.

---

```
const char * strstr ( const char * str1, const char * str2 );
```

Returns a pointer to the first occurrence of `str2` in `str1`, or a null pointer if `str2` is not part of `str1`.

```
str1  
    C string to be scanned.  
str2  
    C string containing the sequence of characters to match.
```

---

```
char *fgets(char *s, int size, FILE *stream);
```

`fgets()` reads in at most one less than `size` characters from `stream` and stores them into the buffer pointed to by `s`. Reading stops after an EOF or a newline. If a newline is read, it is stored into the buffer. A '\0' is stored after the last character in the buffer.

---

Please remember to check whether opening the files is successful. Declare and define all necessary variables. You can assume that each line contains at most 80 characters (including '\n').

*Answer:*

```
#include <string.h>  
#include <stdio.h>  
int wordCount(char * f1name, char * f2name)  
{  
    const int maxLineWidth = 80;  
    FILE * f1ptr;  
    FILE * f2ptr;  
    char oneLine[maxLineWidth + 1];  
    int ece264Count = 0;  
    char * linePtr;  
    f1ptr = fopen(f1name, "r");  
    f2ptr = fopen(f2name, "w");  
    if ((f1ptr == 0) || (f2ptr == 0))  
    {  
        printf("open file fail\n");  
        return -1;  
    }
```

```

while (feof(f1ptr) == 0)
{
    fgets(oneLine, maxLineWidth, f1ptr);
    oneLine[maxLineWidth] = '\0';
    linePtr = oneLine;
    do
    {
        linePtr = strstr(linePtr, "ECE 264");
        if (linePtr != 0)
        {
            ece264Count++;
            linePtr++;
        }
    } while (linePtr != 0);
}
fprintf(f2ptr, "%d\n", ece264Count);
fclose (f1ptr);
fclose (f2ptr);
return 0;
}

/* main function is not necessary for this question */
int main(int argc, char * argv[])
{
    if (argc < 3)
    {
        printf("need two file names\n");
        return -1;
    }
    wordCount(argv[1], argv[2]);
    return 0;
}

```