CSE 220 - C Programming
Fall 2015

Review of Final Exam
The END

• That's the end of CSE220 Fall 2015
  – It has been a great experience to me
  – Thank all of you for your support and suggestions

• Good luck with your finals
Example (I)

Write a recursive function that locates the largest element of an array by recursion:

\[
\text{max}(\begin{array}{cccc}
4 & 5 & 9 & 2 \\
\end{array})
\]

\[
= \text{max}(4, \text{max}(\begin{array}{cccc}
5 & 9 & 2 & -1 \\
\end{array}))
\]

Recursion
Example (I)

Write a recursive function that locates the largest element of an array by recursion

```c
int maxRecursive(int a[], int size)
{
    int result;

    if(size == 1)
        result = *a;
    else
    {
        result = maxRecursive(a+1, size-1);
        result = (*a >= result) ? (*a) : result;
    }

    return result;
}
```
Write a recursive function that locates the largest element of an array by recursion

```
int main()
{
    int arr[] = {4, 5, 9, 2, -1};
    printf("The maximum number is: %d\n", maxRecursive(arr, 5));
    return 0;
}
```
Example(II)

```c
int x[4] = {12, 20, 39, 43}, *y;

y = &x[0]; // y points to the beginning of the array

printf("%d\n", x[0]); // outputs 12
printf("%d\n", *y); // also outputs 12
printf("%d\n", *y+1); // outputs 13 (12 + 1)
printf("%d\n", (*y)+1); // also outputs 13
printf("%d\n", *(y+1)); // outputs x[1] or 20

y+=2; // y now points to x[2]
printf("%d\n", *y); // prints out 39

*y = 38; // changes x[2] to 38
printf("%d\n", *y-1); // prints out x[2] - 1 or 37

*y++; // sets y to point at the next array element
printf("%d\n", *y); // outputs x[3] (43)

(*y)++; // sets what y points to to be 1 greater
printf("%d\n", *y); // outputs the new value of x[3] (44)
```
Example (III)

Write a recursive function that takes an array of numbers and returns the sum of its elements.

\[ \text{sum}(4, 5, 9, 2, -1) = 4 + \text{sum}(5, 9, 2, -1) \]

Recursion
Example (III)

Write a recursive function that takes an array of int and returns the sum of its elements.
Example (III)

Write a recursive function that takes an array of floats and returns the sum of its elements.

```c
int sumRecursive(int a[], int size)
{
    int sum;

    if(size == 1)
        sum = *a;
    else
        sum = *a + sumRecursive(a+1, size-1);

    return sum;
}
```
Example (IV)

Explain what happens in every step:

```c
int a, b = 99, *p;
a = &b;
p = &b;
printf("%d", a);      // print 129119392
printf("%d", p);      // print 129119392
printf("%d", a+1);    // print 129119393
printf("%d", p+1);    // print 129119396
printf("%d", *p);     // print 99
printf("%d", *a);     // error: invalid type argument of
                     // unary '*' (have 'int')
```
Example (V)

Write a recursive function that reverse a string.
Write a recursive function that reverse a string.

```
#define MAX 100

char* getReverse(char str[]){
    static int i=0;
    static char rev[MAX];
    if(*str){
        getReverse(str+1);
        rev[i++] = *str;
    }
    return rev;
}
```
Example (V)
Write a recursive function that reverse a string.

\[
\begin{array}{cccccc}
\text{c} & \text{s} & \text{e} & \text{2} & \text{2} & \text{0} \\
\end{array}
\]

\[
\begin{array}{c}
i = 0 \\
cse220 \\
i = 0 \\
se220 \\
i = 0 \\
e220 \\
i = 0 \\
220 \\
i = 0 \\
20 \\
i = 0 \\
0 \\
\end{array}
\]
Write a recursive function that reverse a string.

Example (V)

*str is ‘0’
Example (V)
Write a recursive function that reverses a string.

```
0 2
```
Write a recursive function that reverse a string.

Example (V)

*str is ‘2’
Example (V)

Write a recursive function that reverse a string.


c  s  e  2  2  0

\[ i = 4 \]
\[ \text{cse220} \]

\[ i = 4 \]
\[ \text{se220} \]

\[ i = 4 \]
\[ \text{e220} \]

\[ *\text{str} \text{ is 'e'} \]

\[ 0 2 2 2\ e \]
Example (V)

Write a recursive function that reverse a string.

```
0 2 2 c se220
```

```
i = 5
se220
```

```
*str is ‘s’
```

```
ic | s | e | 2 | 2 | 0
---|---|---|---|---|---
```
Example (V)

Write a recursive function that reverse a string.

| c | s | e | 2 | 2 | 0 |

\[ i = 6 \]
\[ \text{cse220} \]

*str is ‘c’
Write a recursive function that reverse a string.

Example (V)
Write a C code that takes as input a string (char *str) and returns the number of punctuation characters (‘?’, ‘!’, ‘,’ ‘.’) in it.

```c
#include <stdio.h>

int processString(char *str);

int main()
{
    int max = 100;
    int counts;
    char mystring[max];
    scanf("%s", mystring);
    counts = processString(mystring);
    printf("Count is %d \n\n", counts);
    return 0;
}

int processString(char *str)
{
    int counter = 0;
    while (*str)
    {
        if (*str == '?' || *str == '!' || *str == ',' || *str == '.')
        {
            counter = counter + 1;
        }
        str++;
    }
    return counter;
}
```
What is the output of the following C program:

```c
#include <stdio.h>

int a;

void addOne(void)
{
    a++;
    printf("AddOne(): a = %d\n", a);
}

void removeOne(int a)
{
    b = a - 1;
    printf("Remove(): b = %d\n", b);
}

void swap(int a, int *b)
{
    int temp = a;
    a = *b;
    *b = temp;
    printf("swap(): b = %d\n", *b);
}

void printGlobal_a(int positionIndicator)
{
    printf("GLOBAL %d: a = %d\n", positionIndicator, a);
}

int main()
{
    a = 5;

    int b = 20;
    printGlobal_a(1);
    if (b > 15)
    {
        int a = 53;
        printGlobal_a(2);
        removeOne(b);
        addOne();
        printf("Inside if block: a = %d\n", a);
        printGlobal_a(3);
    }
    printGlobal_a(4);
    printf("Right after if block: a = %d, b = %d\n", a, b);
    swap(a, &b);
    printGlobal_a(5);
    printf("Final values: a = %d, b = %d\n", a, b);
    return 0;
}
```

GLOBAL 1: a = 5
GLOBAL 2: a = 5
Remove(): b = 19
AddOne(): a = 6
Inside if block: a = 53
GLOBAL 3: a = 6
GLOBAL 4: a = 6
Right after if block: a = 6, b = 20
swap(): b = 6
GLOBAL 5: a = 6
Final values: a = 6, b = 6
Explain what happens in every step:

double a[5] = {0.0, 10.0, 20.0, 30.0, 40.0}, *p, *q;

    p = a;
    Set p to point to the 1st element of the array (a[0])

    q = p+1;
    Set q to point to the second element of the array, a[1] (one element after p)

    a[0] = 5.0;
    Change the value of the first element of a to 5.0

    *p = 7.0;
    Change the value p points to (a[0]) to 7.0

    *q = 11.0;
    Change the value q points to (ie a[1]) to 11.0

    q = &a[4];
    Set q to point to a[4]

    printf("%d\n", q - p);
    Print the different between the values of q and p which is 4 (address of a[4] – address of a[0])