CS 171 Computer Programming 1

Winter 2016

Lab 6 – Advanced Functions: Recursion

NAME: LAB:

Question 1: Trivial Tuples (3pts)

In mathematics, a *tuple* (sometimes called a permutation with repetition) is an ordered sequence of elements chosen from some set. Suppose, for example, that our set is {0, 1, 2}. The tuples of length 3 (or 3-tuples) are, 000, 001, 002, 010, 011, 012, 020, 021, 022, 100, 101, 102, 110, 111, 112, 120, 121, 122, 200, 201, 202, 210, 211, 212, 220, 221, and 222. If you noticed that there are $27 = 3^3$ such tuples, that's no accident. For the first position, we have 3 choices, as do we for the second and third positions. So the total number of possible sets of choices we have is 3x3x3.

In this lab, you will be creating a program that serves as a general tuple generator. For the first step, we're going to generate a tuple of length 1. As you might guess, that's trivially easy; we just list the elements of the set. To do this you're going to write a function that takes two character values as parameters and prints the 1-tuples to the standard output. The set from which the elements of the tuple are chosen is the range of characters given as parameters. For example, if the starting parameter is 'a' and the ending parameter is 'c', then the set will be {a, b, c}. The function has no output as far as the rest of the program is concerned; only user output.

Given this descriptions, give a **prototype** for your tuple generating/printing function in your answer sheet.

Answer:

```
void tuplegenerator (char a, char c)
```

Question 2: Loop Design (5pts)

The next step is to determine how your function will generate the tuples. Because it's going to be repeating some action, you'll need a loop (for now anyway!). The first question you might as yourself is: “what kind of loop makes the most sense in this situation?”. As always, be sure you can fill in the blank: This loop is done once for each _____.

In your lab solution, put the **full loop (including the loop body)** for generating **1-tuples**. You may assume that you already have values from the user for start and stop and that start <= stop. If you choose your loop well, you should be able to do this in 2 lines of active code (not counting declarations and lines with only punctuation).

Note: You only need to show the loop, in the next question you'll actually put it into a function.

Answer:

```
for( char x=a ;x<=b; x++)
  {
     cout<<x;
   }
```

Question 3: Making a Program (10pts)

Now put the pieces together. You should have a program that has two functions, *main*() and the tuple function you've just designed. *main*() should prompt the user for the starting and ending characters of the range that will make up the set of characters. It then calls your tuple function to print the tuples.

A few things:

1. Now if start > stop, then the list should be empty {}
2. If the user entered 'C' and 'H' for start and stop, respectively, then the output should be *exactly* (with brackets):

   {C, D, E, F, G, H}

   Hint: You'll likely have to use some conditional branches now in your loop to get this to look correct.

Once you have this working correctly, **paste your tuples function into your answer sheet** (for now we just want this function)**:**

Answer:

```
void printTuples(char a, char b)
{
   cout<<"{";
for( char x=a ;x<=b; x++)
   {
      cout<<x;
      if(x!=b)
```

```
        cout<<",";
      else
        cout<<"}";
   }
}
```

Question 4: Tuples of Length 2 (5pts)

Modify your tuple function to produce 2-tuples (also called pairs). For a range of size of `n`, you'll be outputting n^2 tuples.

For example if the user entered 'C' and 'E' for start and stop, respectively, then the output should be *exactly* (with brackets):

{CC, CD, CE, DC, DD, DE, EC, ED, EE}

Like for the previous question, cut and paste your new tuple function into the answer sheet when it is completed and tested.

Answer:

```
// Joseph Mulray
#include <iostream>
using namespace std;
```

```
void printTuples(char a, char b)
{
   cout<<"{";
for( char x=a ;x<=b; x++)
   {
      for( char j=a ;j<=b; j++)
      {

      cout<<x;
      cout<<j;
      if(x<b || j<b)
         cout<<",";
      else
         cout<<"}";

      }

   }
}

int main()
{
char a;
char b;

cout<<"Enter the first character: ";
```

```
cin>>a;

cout<<"Enter the second characrer: ";
cin>>b;



printTuples(a,b);

return 0;




}
```

Question 5: Tuples of Length 3 (3pts)

If you got your program working with 2-tuples, you will likely have a pair of nested loops in your tuple function. There's a good chance you designed it so that the outer loop is done once for each element of the set in the first position, and the inner loop is done once for each element of the set in the second position.

As you might guess, for this step, you are to expand your program further and have it generate 3-tuples.

Like for the previous question, cut and paste your new tuple function into your answer sheet when it is completed and tested.

Answer:

```
#include <iostream>

using namespace std;


void printTuples(char a, char b)

{

   cout<<"{";

for( char x=a ;x<=b; x++)

   {

      for( char j=a ;j<=b; j++)

      {

         for( char n=a ;n<=b; n++)

         {

      cout<<x;

      cout<<j;

      cout<<n;

      if(x<b || j<b || n<b)

         cout<<",";
```

```
        else
           cout<<"}";
           }

        }

    }
}

int main()
{
char a;
char b;

cout<<"Enter the first character: ";
cin>>a;

cout<<"Enter the second characrer: ";
cin>>b;



printTuples(a,b);

return 0;
```

}

Question 6: k-tuples (2pts)

If I were to ask you to modify your program to generate tuples of some fixed length, say k , **how many levels of nesting would your loops have** (relative to k)? I.e. how many **for** statements would you have?

Answer:

K tuples

Question 7: Another Program Modification (2pts)

In anticipation of the remaining steps of the lab, modify your program as follows:

- Ask the user for the size of the tuples desired
- Add two new parameters to your tuple generating function prototype
    - The size of the desired tuples
    - A string

Paste your new function **prototype** below:

Answer:

void printTurple(char a, char b, int size, string str)

Question 8: Another way to define tuples (2pts)

Now let's get into some more formal mathematical notation…..

Let's denote the set of k-tuples from a set Σ by S_Σ^k. So if $\Sigma = \{a, b, c\}$ then

$$S_\Sigma^2 = \{aa, ab, ac, ba, bb, bc, ca, cb, cc\}$$

Next, let's say we have an element of Σ called α. From here on, we're going to call such an element a *symbol* so we don't have to keep saying "element of Σ". Then αS_Σ^k will be all the tuples that start with α and end with a k-tuple. So in our example,

$$aS_\Sigma^2 = \{aaa, aab, aac, aba, abb, abc, aca, acb, acc\}$$

In your answer sheet, enter the tuples for bS_Σ^2 (note: no code for this one!)

Answer:
$bS_\Sigma^2 = \{bbb, bbc, bbd, bcb, bcc, bcd, bdb, bdc, bdd\}$

Question 9: Recursive tuples (5pts)

Notice that both aS_Σ^2 and the bS_Σ^2 that you just entered (your previous answer) are both 3-tuples. In fact if we take all of the sets aS_Σ^2 , bS_Σ^2 , and cS_Σ^2 , then we get the same thing as S_Σ^3. In effect, we can say that to create the set of `n`-tuples, we can start with the set of `(n-1)` -tuples and prepend (put before) each symbol to a whole set of the `(n-1)` -tuples. Or more formally

$$S_\Sigma^n = \{\alpha S_\Sigma^{n-1}, for\ each\ \alpha \in \Sigma\}$$

In English, what we're saying is that if we know how to make tuples of one size, we can make tuples of the next size up, by one-at-a-time pasting a symbol onto all the tuples of the smaller size. Defining an operation such as generating `n` -tuples in terms of a simpler version of itself (generating `n-1` -tuples) is an example of a technique called *recursion*.

Given $\Sigma = \{1, 2, 3, 4\}$ list the tuples for S_Σ^1 in your answer sheet. Then apply the recursive technique described here to generate S_Σ^2 and also put that in your answer sheet. Again, no actual programming here!

Answer:
{1,2,3,4}
`{11,12,13,14,21,22,23,24,31,32,33,34,41,42,43,44}`

Question 10: Starting the Recursive Process (5pts)

Now we'd like to turn this into a useful recursive algorithm. Let's say we have a tuple that we'll call `P`. As you might expect, PS_Σ^n is a `P` followed by each of the n-tuples. Combining this idea of prefixes with recursion gives us the statement:

$$PS_\Sigma^n = \{P\alpha S_\Sigma^{n-1}, for\ each\ \alpha \in \Sigma\}$$

Which basically says that we can generate a set of tuples by taking the prefix (P), tacking a symbol onto the end of it (α) and generating all the tuples one size smaller (S_Σ^{n-1}). This will in turn, cause us to tack another symbol onto the prefix and generate tuples two sizes smaller. Etc.., etc.. We keep "drilling down" this way until we can't go any farther. After all our

recursion needs a *base case*. When we hit this (which in this case is when we're generating tuples of size n=0), our prefix is an n-tuple that we want to print out.

The basic algorithm can be written as follows:

To print all `n` -tuples with prefix `P`:

- If `n=0` , print `P` (base case)
- Otherwise,
    - For each symbol α,
        - Recursively create all (`n-1`)-tuples with prefix $P\alpha$

Here's an example trace of this for $\Sigma = \{1,2\}$ and n=2

- Initially call `printTuples` with `n=2` and the current tuple (prefix) as empty " ".
- For each $\alpha \in \Sigma$ ….
    - For $\alpha = 1$ call `printTuples` with n=1 and current tuple as "1"
        - For each $\alpha \in \Sigma$ ….
            - For $\alpha = 1$ call `printTuples` with `n=0` and current tuple as "11"
                - Now `n=0` so just print out the current tuple (which is "11")
            - For $\alpha = 2$ call `printTuples` with `n=0` and current tuple as "12"
                - Now `n=0` so just print out the current tuple (which is "12")
    - For $\alpha = 2$ call `printTuples` with `n=1` and current tuple as "2"
        - For each $\alpha \in \Sigma$ ….
            - For $\alpha = 1$ call `printTuples` with `n=0` and current tuple as "21"
                - Now `n=0` so just print out the current tuple (which is "21")
            - For $\alpha = 2$ call `printTuples` with `n=0` and current tuple as "22"
                - Now `n=0` so just print out the current tuple (which is "22")

In the answer box below provide an example trace for $\Sigma = \{\mathbf{a}, \mathbf{b}, \mathbf{c}\}$ and n=2 (no code here)

Answer: `{aa,ab,ac,ba,bb,bc,ca,cb,cc}{a,b,c}`

Question 11: Implementing the Recursive Method (10pts)

Ok, hopefully from the previous traces you got a feel for how this can be written. Modify your program to use this recursive technique to generate tuples.

When you have it debugged, cut and paste your **entire program** into your answer sheet.

Answer:

```
// Joseph Mulray Lab 6
#include <iostream>
#include <string>
using namespace std;

void printTuples(char a, char b, int n, int currentstring)
{



   if(n!=0)
   {
      cout<<"{";
      for(char x=a ;x<=b; x++)
      {
      cout<<x;
      if(x!=b)
         cout<<",";

      else
         cout<<"}";
      }
   }
   else
   {
    printTuples(a,b,n-1, a+currentstring);
   }
```

```
}
int main()
{
char a;
char b;
int n;

cout<<"Enter the number of tuples: " ;
cin>>n;

cout<<"Enter the first character: ";
cin>>a;

cout<<"Enter the second characrer: ";
cin>>b;

int currentstring = n;

printTuples(a,b, n,currentstring);

return 0;



}
```