# CS 221 C and Systems Programming
# Assignment 4

**Due at 11:59 pm on October 12, 2015**

## 1 Arrays as Function Parameters (30 marks)

Write a recursive C function to compute the inner product of two arrays:

```
double inner_product(double *a, double* b, int n);
```

Both `a` and `b` are pointer variables for arrays of size `n`. The function should return: `a[0]*b[0]+a[1]*b[1]+···+a[n-1]*b[n-1]`. Use pointer arithmetic, not indexing, to visit array elements.

Use the following `main` function to test your code:

```
main()
{
 double v[] ={2.3, 6.0, 1.2, 0.7, 9.4, 5.1, 0.2, 4.4, 2.3, 0.01};
 printf(``%f \n'',inner_product(v,v,10));
 printf(``Expected:  182.2801\n'');
}
```

Name your file `product.c`.

## 2 Dynamic Memory Management (40 marks)

Implement the following three string functions using pointer arithmetic:

- `char* twice(const char* s)`: this function returns a string that contains two copies of the input string.
- `char* reverse(const char* s)`: this function returns the reverse of the input string.
- `char* drop(const char* s, char c)`: this function returns the input string with all occurrence of character `c` removed from the string.

In this implementation, define a local pointer variable to store the result. The local variable is a `char *` which points to a dynamically allocated memory. The function returns the local pointer variable as the return value.

```
char* twice(const char* s)
{
   char* result;
   result = (char *) malloc(sizeof (char)* size);
```

```
  // you need to figure out the size of the array needed to store the result
  if(result == NULL) return NULL; // heap exhausted

  // your code here

  return result;
}
```

Use the following `main` function to test your code. Add code to `main()` to `free` the memory allocated in the functions, so that your program does not have memory leakage at run time.

```
main()
{
   char s[] = "quick";
   printf("%s\n", twice(s);
   char u[] = "jumps over";
   printf("%s\n", reverse(u);
   char v[] = "lazy dog";
   printf("%s\n", drop(v, 'o');
}
```

Name your file `string.c`.

## 3 Command-Line Arguments (30 marks)

Write a C program that compute its command-line arguments, according to the given flag. The program supports 3 types of flag: `-a` for addition, `-s` for subtraction and `-m` for multiplication. The command-line arguments are integers. For example,

```
a.out -a 8 2 4
14
a.out -s 10 1 9 3 3
-6
a.out -m 3 1 2 4
24
```

Name your file `command.c`. Hint: Use the `atoi` function to convert each command-line argument from string to integer.