Pre-test for C (for COP 3502, Monday, August 19th) – SOLUTION

1) Write a segment of code that prints out the following sequence of numbers: 2, 5, 8, …, 299999, printing one number per line.

```
int i;

for (i=2; i<300000; i+=3)

    printf(“%d\n”, i);
```

2) Complete the function below that takes in an integer array and its length and returns the average of the values in the array.

```
double average(int array[], int n) {

    double sum = 0;

    int i;

    for (i=0; i<n; i++)

        sum += array[i];

    return sum/n;

}
```

3) Complete the segment of code below so that it prints out the name that comes first alphabetically of the 100 names entered by the user. Assume that all names entered contain only uppercase letters. You may use the string functions strcmp and strcpy.

```
char names[100][20];

printf(“Please enter your 100 names.\n”);

int i;

for (i=0; i<100; i++)
```

```
    scanf(“%d”, &names[i]);

int best = 0;
for (i=1; i<100; i++)
    if (strcmp(names[i], names[best]) < 0)
        best = i;

printf(“The first name is “, names[best]);
```

4) Using the following struct definition to store a single playing card, write a segment of code that declares an array of size 52 of the struct and fills it with one of each standard playing card. For ease, you may use the constant arrays declared below .

```
const char SUITS[] = "CDHS";

const char KINDS[] = "A23456789TJQK";


struct card {

    char suit;

    char kind;

};
```

```
struct card deck[52];

int i,j;

for (i=0; i<4; i++) {

    for (j=0; j<13; j++) {

        deck[13*i+j].suit = SUITS[i];

        deck[13*i+j].kind = KINDS[j];

    }

}
```

5) Given the struct used to define a linked list below, write a function that, given a pointer to the front of a linked list, returns the last value stored in the list. You are guaranteed that the input pointer is pointing to a list of at least size 1.

```
struct node {

    int data;
```

```c
    struct node* next;
};

int getLastValue(struct node* front) {

    while (front->next != NULL)
        front = front->next;

    return front->data;
}
```