1(a). Take $i = 0, j = 1$. The first iteration of the outer loop compares $n - j = 5 - 1 = 4$ times. No element $A[j]$ satisfies $A[j] < A[small]$, so there are no swaps. The next iteration of the outer loop, with $i = 1$, sets $j = 2$ and we perform $5-2 = 3$ comparisons. Again there is no swap. With the next iterations, we compute another 2, then 1. Together there are $4 + 3 + 2 + 1 = 10$ comparisons with 0 swaps.

1(b). Since there are the same number of elements, then there are again 10 comparisons. The function performs 2 swaps.

1(c). There are 10 comparisons with 2 swaps.

2(a). For $n = 0$ or $n = 1$, there are 0 comparisons.

For $n > 1$, one iteration of the outer loop performs $n - j$ comparisons. The next iteration increments j then performs $n - j$ comparisons. When $n - j = 1$, the last comparison is performed. So $n - 1$ is the upper bound. The comparisons occur unconditionally. Hence the minimum and maximum are equal and are determined by

$$\sum_{j=1}^{n-1} n - j.$$

2(b). For $n = 0$ or $n = 1$, there are 0 swaps.

For $n > 1$, the maximum number of swaps is $n - 1$, and the minimum number of swaps is 0.

3.

```
BOOLEAN lexless(LIST A, LIST B)
{
    if (A == NULL && B != NULL) return TRUE;
    else if (B == NULL) return FALSE;
    else if (A->element < B->element) return TRUE;
    else if (B->element < A->element) return FALSE;
    else return lexless(A->next, B->next);
}
```

4.

```
BOOLEAN lexless(LIST A, LIST B)
{
    if (A == NULL && B != NULL) return TRUE;
    else if (B == NULL) return FALSE;
    else if (normalizecase(A->element) < normalizecase(B->element))
        return TRUE;
    else if (normalizecase(B->element) < normalizecase(A->element))
        return FALSE;
    else return lexless(A->next, B->next);
}

char normalizecase(char c)
```

```
{
    int difference;
    difference = 'a' - 'A';
    if (c >= 'A' && c <= 'Z') return c + difference;
    else return c;
}
```

5. Selection sort performs $\sum_{j=1}^{n-1} n - j$ comparisons and 0 swaps.

6.

```
void SelectionSort(struct STUDENT A[], int n)
{
    int i, j, small;
    struct STUDENT temp;
    for (i = 0; i < n-1; i++) {
        small = i;
        for (j = i+1; j < n; j++)
            if (A[j].studentID < A[small].studentID)
                small = j;
        temp = A[small];
        A[small] = A[i];
        A[i] = temp;
    }
}
```

7.

```
void SelectionSort(T A[], int n)
{
    int i, j, small;
    T temp;
    for (i = 0; i < n-1; i++) {
        small = i;
        for (j = i+1; j < n; j++)
            if (lt(key(A[j]), key(A[small]))) /* assume lt, key exist */
                small = j;
        temp = A[small];
        A[small] = A[i];
        A[i] = temp;
    }
}
```

8.

```
void SelectionSort(int A[], int n)
{
    int *i, *j, *small, temp;
    for (i = A; i < A + n-1; i++) {
```

```
        small = i;
        for (j = i+1; j < A + n; j++)
            if (*j < *small)
                small = j;
        temp = *small;
        *small = *i;
        *i = temp;
    }
}
```

9.

```
void SelectionSort(struct STUDENT *A[], int n)
{
    int i, j, small;
    struct STUDENT *temp;
    for (i = 0; i < n-1; i++) {
        small = i;
        for (j = i+1; j < n; j++)
            if (A[j]->studentID < A[small]->studentID)
                small = j;
        temp = A[small];
        A[small] = A[i];
        A[i] = temp;
    }
}
```

10.

```
main()
{
    SelectionSort(A,n);
    printdistinct(A,n);
}

void printdistinct(int A[], int n)
{
    int i, recent;
    if (n > 0) {
        printf("%d\n", A[0]);
        recent = A[0];
    }
    for (i = 1; i < n; i++) {
        if (A[i] != recent)
            printf("%d\n", A[i]);
        recent = A[i];
    }
}
```

11.

a) $\sum_{i=1}^{189} 2i - 1 = 1 + 3 + 5 + \cdots + 377$

b) $\sum_{i=1}^{n/2} (2i)^2 = 4 + 16 + 36 + \cdots + n^2$

c) $\prod_{i=3}^{k} 2^i = 8 + 16 + 32 + \cdots + 2^k$

12. Suppose $small = i$. Then `A[small]` has the value $A[i]$. Thus `temp` is assigned $A[i]$, then `A[small]` is assigned $A[i]$, and `A[i]` is assigned $A[i]$. Since `A[small]` and `A[i]` are assigned to their own values, then there is no mutation that is performed, so `A` remains unchanged.