SANDRA PAPO
1041213584

Q1 a. (1) 7.500000
(2) 7.000000
(3) 6.400000
(4) -6.000000
(5) -14.000000

(b) (1) int a = rand()%100+1;
(2) int b = rand()%10;
(3) int c = rand()%113 + 1000;
(4) int d = rand()%3 - 1;
(5) int e = rand()%15 - 3;

(c) (1) int a = (rand()%5 + 1) * 2;
(2) int b = ((rand()%5 + 1) * 2) + 1;
(3) int c = (rand()%5 + 1) * 4 + 2;

(d) (1) int counts[10] = {0};
(II) for(size_t i=0; i<15; i++) bonus[i] += 1;
(iii) for(size_t i=0; i<12; i++)
scanf("%f", &monthlyTemperatures[i]);
(IV) for(size_t i=0; i<5; i++)
printf("%d\t", bestScores[i]);

(e) (I) False; we specify the index/location not value
(II) True.
(III) False: int p[100];
(IV) False: you only need 1 statement int arr[15]={0};
(V) True:
(VI) True:

(f) (I) int t[2][5];
(II) 2 rows, 5 columns
(III) t[0][2], t[1][2], t[2][2]
(IV) t[1][2]=0;

(v) `int t[2][5] = {0};`

(vi)
```
for (size_t i = 0; i < 2; i++){
    for (size_t j = 0; j < 5; j++)
        t[i][j] = 0;
}
```

(vii)
```
for (size_t i = 0; i < 2; i++){
    for (size_t j = 0; j < 5; j++)
        scanf("%d", &t[i][j]);
}
```

(viii)
```
int min = t[0][0];
for (size_t i = 0; i < 2; i++){
    for (size_t j = 0; j < 5; j++){
        if (t[i][j] < min)
            min = t[i][j];
    }
    printf("The min value is %d", min);
}
```

(ix)
```
for (size_t i = 0; i < 5; i++)
    printf("%d\t", t[0][i]);
```

(x)
```
int sum = 0;
for (size_t i = 0; i < 2; i++)
    sum += t[i][3];
```

(xi)
```
printf("\t 1 \t 2 \t 3 \t 4 \t 5");
for (size_t i = 0; i < 2; i++){
    printf("\n%d", i+1);
    for (size_t j = 0; j < 5; j++)
        printf("\t%d", t[i][j]);
}
```

Q2. This function recursively iterats through an array of int's passed in and prints out each element in the array seperated by spaces. The output is as follows.

5 4 7 9 0 6 2 1 3 8

The function takes the input paramaters of an array of integers, a starting index and the size of the array. It effectively starts at the startIndex specified when the function is first called in the main, then recursively iterates through and prints every element thereafter until it reaches the end of the array.