Anish Laddha, Period 1

Chapter 6

Page 292

R6.1, 2, 6, 10, 11, 13 to 18, 23, 25

1.

a.

double values[10];

for(double i = 0; i<10; i++)

{

values[i] = i;

}

b.

double values[10];

for(double i = 0; i<10; i++)

{

values[i] = i\*2;

}

c.

double values[10];

for(double i = 0; i<10; i++)

{

values[i] = i\*i;

}

d.

double values[10];

for(double i = 0; i<10; i++)

{

values[i] = 0;

}

f.

double values[10];

for(double i = 0; i<10; i++)

{

if (i%2.0 == 0)

{

values[i] = 0;

}

else

{

values[i] = 1;

}

}

g.

double values[10];

for(double i = 0; i<5; i++)

{

values[i] = i;

}

for(double i = 5; i<10; i++)

{

values[i] = i-5;

}

2.

1. 25
2. 13
3. 12
4. 22
5. 0, infinite loop and index doesn’t change
6. 25
7. 12
8. -1

6.

The array is 10 elements long, since its 0-indexed, the last index is 9. However, the loop tries to do something at the tenth index which doesn’t exist. One way to fix this would be to make i = 0 at the beginning, and making the conditioning i<10. Another way to fix it would be to make the operation *values[i-1] = i\*i;*.

10.

|  |  |
| --- | --- |
| i | out |
| 0 | “0” |
| 1 | “ | 1” |
| 2 | “ | 4” |
| 3 | “ | 9” |
| 4 | “ | 16” |

11.

|  |  |
| --- | --- |
| i | matches |
| 110 | 110 |
| 90 | 110 |
| 100 | 110, 100 |
| 120 | 110, 100, 120 |
| 80 | 110, 100, 120 |

13.

|  |  |
| --- | --- |
| i | values |
| 110 | 110 |
| 90 | 90 |
| 100 | 120 |
| 120 | 80 |
| 80 | del |

14.

a. int sort\_decrease(double array[],int size);

b. void print(double array[],int size,string s);

c. int count(double array[],int size,int value);

d. int new\_array(double array[],int size);

e. int new\_array(double array[],int size,int value);

15.

int n = sizeof(x\_vals)/sizeof(x\_vals[0])

sort(x\_vals, x\_vals + n);

sort(y\_vals, x\_vals + n);

int large\_x = x\_vals[n-1];

int small\_x = x\_vals[0];

int large\_y = y\_vals[n-1];

int small\_y = y\_vals[0];

//bounds^^^^

16.

int n = sizeof(values)/sizeof(values[0])

sort(values, values+n);

double final\_score;

for (int i = 1; i<n; i++)

{

final\_score+=values[i];

}

17.

Repeat from(0 to ½ num of elements):

Remove first, move each one over to the right one, insert first at end

It requires us to move n elements by 1, ½ n times. This gives us a complexity of ½ n^2. Swapping requires a complexity of ½ n or n.

18.

Create a new list of coins. From the first list, iteratively move through it. When you get a coin, see if it exists in the new list. If not, add it there and add a paperclip to represent it. If it already exists, add another paperclip to represent it.

23.

for (0 to row num (excl))

{

if row == 0:

Set to -1

else if row == num-1:

Set to -1

else:

Set first and last pos in row to -1

}

25.

a.

Use ‘==’ to compare.

b.

Swap function can swap 2 vectors of the same datatype. Use a constructor if the second array does not already exist.

c.

Get size of array, use it to create for loop where you set the values of arr[i] to 0

d.

Run the clear() function which effectively resets it.