Justin Ma

Problem 1:

a)

```
int main()
{
    int arr[4] = {0, 1, 2, 3};
    int* ptr = arr;

    *ptr = 1;
    *(ptr + 1) = arr[ 0 ] * 10;
    ptr += 2;
    ptr[0] = arr[ 1 ] * 10;
    ptr[1] = 1000;
    ptr += 1;

    while (ptr >= arr)
    {
        cout<<" "<<*ptr;
        ptr--;
    }
    cout << endl;
    return(0);
}
```

b)
The function needs an ampersand in the parameters to be called by reference. This makes it so that whatever changes were made in the function are carried outside the function. It should look like below:

```
void findLastZero(int arr[], int n, int* &p)
{
    p = nullptr;
   for (int k = n - 1; k >= 0; k--)
    {
        if (arr[k] == 0)
        {
            p = arr + k;
            break;
        }
    }
}
```

c)
The main function needs to initialize pointer as a single integer. It should look like below:

```
int main()
{
    int a;
    int* p = &a;
    biggest(15, 20, p);
    cout << "The biggest value is " << *p << endl;
    return( 0 );
}
```

d)
The functions need to convert the str1 and str2 into pointers. Otherwise, str1 and str2 would only be the first character of the c string. The solution is below:

```
bool match(const char str1[], const char str2[])
{
    bool result = true;
```

```
    while (*str1 != 0  &&  *str2 != 0)
    {
        if (*str1 != *str2)
        {
            result = false;
            break;
        }
        str1 ++;
        str2 ++;
    }
    if (result)
    {
        result = (*str1 == *str2);
    }
    return( result );
}
```

e)
The problem lies in the computeFibonacciSequence function. arr[8] is defined inside the function so after the function is executed, the array will disappear. The pointer it returns will be pointing to nonsense.

Problem 2:

1.f
2.g
3.a
4.b
5.d
6.c
7.b
8.e
9.h

Problem 3:

Array (array [0] which points to the first number of the array which is 5) and &array[2] (points to the 3rd number in the array which is 4) are put in the minimart function. This returns the smaller number which is 4. The pointer ptr is initialized pointing to the 3rd number in the array. In the next line, the item in the next position in the array becomes 9, so 17 becomes 9. Next, the pointer points to the item 2 places down, so it now points to 22. In the next line, whatever the pointer is pointing to becomes -1, so 22 becomes -1. Then the second item of the array becomes 79.
The first output line firsts prints out a string. Then, it outputs the position of the 6th item of the array minus the position of the pointer. The difference between them is one, so the first cout line should look like:

diff=1

There is a new line after that. Then the pointer to the first item is swapped with the pointer to the second item of the array. The next line swaps the values of the first item and the third item, so the first items becomes 4 and the third item becomes 5. The next few lines print out each integer of the array starting with the first one. After each integer, there is a new line. The output should look like:

4
79
5
9
-1
19

Problem 4:

```
void deleteCapitals(char n[])
{
    while(*n != 0)
    {
        char *yea;
        yea = n;
        if (*n == 'A' || *n == 'B' || *n == 'C' || *n == 'D' || *n == 'E' || *n == 'F' || *n
== 'G' || *n == 'H' || *n == 'I' || *n == 'J' || *n == 'K' || *n == 'L' || *n == 'M' || *n ==
'N' || *n == 'O' || *n == 'P' || *n == 'Q' || *n == 'R' || *n == 'S' || *n == 'T' || *n ==
'U' || *n == 'V' || *n == 'W' || *n == 'X' || *n == 'Y' || *n == 'Z')
        {
            while (*yea != 0)
            {
                *yea = *(yea+1);
                yea++;
            }
        }
        n++;
    }
}
```