Assignment 6

- Student Name: Yahia Hany Gaber
- Student ID: 231000412

1. Copying the content of an array to another.

```c
#include <stdio.h>

#define loop(start, end) for (int start = 0; start < end; start++)
#define SIZE 9
#define NEWLINE printf("\n");

void copy(int *arr1, int *arr2);

int main() {

  int test[SIZE] = {1, 2, 3, 4, 5, 6, 7, 8, 9};
  int copied[SIZE];
  copy(test, copied);
  int y = 0;
  loop(y, 9) { printf("%d ", *(copied + y)); }
  NEWLINE;
}

void copy(int *arr1, int *arr2) {
  size_t size = sizeof(arr1) / arr1[0];
  for (int x = 0; x <= size; x++) {
    *(arr2 + x) = *(arr1 + x);
  }
}
```

2. Calculating the factorial of a number.

```c
#include <stdio.h>

#define loop(start, end) for (int start = 1; start <= end; start++)
#define NEWLINE printf("\n");

int factorial(int *number);
int main() {
  int test = 5;
  printf("%d", factorial(&test));
  NEWLINE;
}

int factorial(int *number) {
  int factor = 1;
  int y;
  loop(y, *number) { factor *= y; }
  return factor;
}
```

3. Count the number of even and odd numbers in an array.

```c
#include <stdio.h>

#define loop(start, end) for (int start = 0; start < end; start++)
#define SIZE 6
#define NEWLINE printf("\n");

struct parity {
  int even, odd;
};
typedef struct parity S;

S ParityCount(int *arr);
int main() {
  int test[SIZE] = {1, 2, 3, 4, 5, 5};
  S count = ParityCount(test);
  printf("The number of even numbers: %d\nThe number of odd numbers:  %d\n",
         count.even, count.odd);
}

S ParityCount(int *arr) {
  S p;
  p.odd = 0;
  p.even = 0;
  loop(y, SIZE) {
    p.odd += (*(arr + y) % 2) ? 1 : 0;
    p.even += (!(*(arr + y) % 2)) ? 1 : 0;
  }
  return p;
}
```

4. Remove Duplicates from an array.

```c
#include <stdio.h>

#define loop(start, end) for (int start = 0; start < end; start++)
#define SIZE 6
#define NEWLINE printf("\n");

void removedupes(int *arr, int *arr2, int size);
int countdupes(int *arr, int size);
int main() {
  int test[SIZE] = {1, 1, 1, 2, 2, 3};
  int size = countdupes(test, SIZE);
  int nodupes[SIZE - size];
  removedupes(test, nodupes, SIZE);
  loop(z, size) { printf("%d ", *(nodupes + z)); }
  NEWLINE;
}

void removedupes(int *arr, int *arr2, int size) {
  int count = 0;
  loop(y, size) {
    if (*(arr + y) != 0) {
      *(arr2 + count) = *(arr + y);
      count++;
    }
  }
}

int countdupes(int *arr, int size) {
  int count = 0;
  loop(y, size) {
    if (*(arr + y) == *(arr + (y + 1))) {
      *(arr + y) = 0;
      count++;
    }
  }
  return count;
}
```

5. Finding the intersection between two arrays.

```c
#include <stdio.h>

#define loop(start, end) for (int start = 0; start < end; start++)
#define SIZE 6
#define NEWLINE printf("\n");

void intersection(int *arr1, int *arr2, int *inter);
int countelements(int *arr);
void addintersection(int *arr, int *interarr, int *inters);
int main() {

  int test1[SIZE] = {1, 2, 3, 4, 5, 6};
  int test2[SIZE] = {4, 5, 6, 7, 8, 9};
  int inter[SIZE];
  intersection(test1, test2, inter);
  size_t newsize = countelements(inter);
  int interelements[newsize];
  addintersection(test2, inter, interelements);

  loop(m, newsize) { printf("%d ", *(interelements + m)); }
  NEWLINE;
}

void intersection(int *arr1, int *arr2, int *inter) {
  size_t size1 = sizeof(arr1) / *arr1 + 1;
  size_t size2 = sizeof(arr2) / *arr2 + 1;
  loop(x, size1) {
    loop(y, size2) {

      if (*(arr1 + x) == *(arr2 + y)) {

        *(inter + y) += 1;
      }
    }
  }
}

int countelements(int *arr) {

  int counter = 0;
  size_t size = sizeof(arr) / *arr + 1;
  loop(l, size) {
    if (*(arr + l)) {
      counter++;
    }
  }
  return counter;
}

void addintersection(int *arr, int *interarr, int *inters) {

  size_t size = sizeof(arr) / *arr + 1;
  int g = 0;
  loop(z, size) {

    if (*(interarr + z)) {
      *(inters + g++) = *(arr + z);
    }
  }
}
```