CS19003 Programming and Data Structures Lab

Assignment Set 4 **March 28, 2023**

INSTRUCTIONS

1. There are three assignments in this Lab. You need to submit each of the three assignments separately. It is advisable to submit each assignment as you complete it, rather than wait for the end to submit everything.
2. Your source program files must be named exactly as indicated (note that names are case sensitive)
3. Please write a header as indicated in the previous assignment

---

1. **[Filename: set4asg1.c]**

   Working with Arrays. Write a C program that reads an integer N (assume N < 20), then reads N integers into an array, `arr[]`, and invokes the following functions:

   (a) Write the following two functions that return respectively the second maximum and second minimum of the N integers:

   ```
   int second_max(int arr[], int N)
   int second_min(int arr[], int N)
   ```

   For example, if the numbers are 10, 20, 40, 30, 60, then the second minimum is 20 and the second maximum is 40. ***Sorting the array is not needed, and will earn deduction of marks.***

   (b) Write a function, `float frac_above_average(int arr[], int N)`, that computes and returns the fraction of numbers that are above average and returns that as a percentage. For example, if the numbers are 10, 20, 40, 30, 60, then the answer is 20%, and the function returns 20.0.

   (c) Write a function, `int len_long_seq(int arr[], int N)`, that finds and returns the length of the longest sequence of consecutive identical numbers. For example, if the numbers are 30, 20, 10, 10, 40, 40, 40, 10, 10, then the answer returned is 3.

   (d) Write a function, `int len_wrapped_long_seq(int arr[], int N)`, that finds and prints the length of the longest sequence of consecutive identical numbers with wrapping around, that is, when the sequence represents a circular list of numbers. For example, if the numbers are 10, 10, 20, 10, 10, 40, 40, 40, 10, 10, then the answer returned is 4 (corresponding to the sequence consisting of the last two 10s in the array followed by the first two 10s in the array after wrapping around).

   (e) Write a function, `cumseq()`, with suitable parameters and return type, that computes and prints the *right cumulative sequence*. In a *right cumulative sequence*, each number `arr[i]` is replaced with the sum of all elements from `arr[i]` to `arr[N-1]`. For example, if the numbers are 10, 20, 40, 30, 60, then it should print 160, 150, 130, 90, 60.

   The main program calls each of the above function and prints the values returned by the functions.

2. **[Filename: set4asg2.c]**

Inverse permutation. An inverse permutation is a permutation in which each number and the index of the place which it occupies are exchanged. For example, the following are inverse permutations:

```
a[] = { 2, 7, 4, 9, 8, 3, 5, 0, 6, 1 }
b[] = { 7, 9, 0, 5, 2, 6, 8, 1, 4, 3 }
```

If the permutation is in an array `a[]`, then its inverse is the array `b[]` such that `a[b[i]]= b[a[i]]= i.`

Write a C program that reads in an integer N (assume N < 20), followed by a permutation of the integers 0 to N –1, which is stored in an array `A[]`. After this it does the following.

- It checks whether the input sequence of N integers is a valid permutation of the numbers 0 to N –1. If not, it prints `Input is invalid` and terminates.
- It computes the inverse permutation of array, `A[]`, in another array, `B[]`.
- It prints the arrays `A[]` and `B[]`.

3. **[Filename: set4asg3.c]**

Recursion. The ministry of magic produces coins of denomination 3, 5 and 10 respectively. The function, **`canchange(k)`**, returns –1 if it is not possible to pay a value of *k* using these coins. Otherwise it returns the ***minimum*** number of coins needed to make the payment.

For example, **`canchange(7)`** will return –1. On the other hand, **`canchange(14)`** will return 4 because 14 can be paid as 3+3+3+5 and there is no other way to pay with fewer coins.

A code skeleton for the function is given below as a hint. This is not complete, and has missing statements and missing expressions, indicated with question marks.

```
int canchange(int k)
{
   int a= ?? ;
   if (k==0) return 0;
   if ( ?? ) return 1;
   if (k < 3) ??;

  a = canchange( ?? );
  if (a > 0) return ?? ;
```

```
    a = canchange(k – 5);
    if (a > 0) return ?? ;

    a = canchange( ?? );
    if (a > 0) return ?? ;
    ??
}
```

(a) Complete the function and write a main( ) to read an input number, call the function with it, and print the value it returns.

(b) Modify the function of part (a) to write a function to print the change. For example, if we call the function **`printchange(14)`** it should print 3+3+3+5. The function prototype is:

```
int printchange(int k)
```