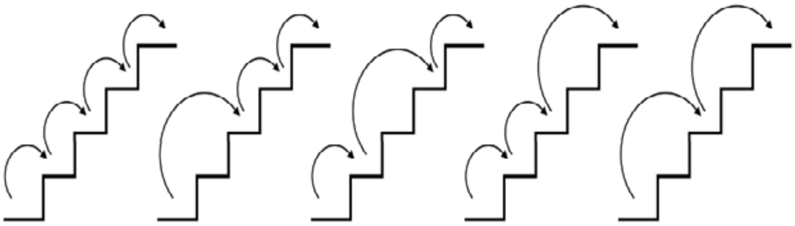
Prog. and Data Structures Lab (Section -1) Spring 2021

Assignment Set 4

1. **Frog Climbing Stairs.** A frog is climbing a staircase, starting from Stair-0, with the hope of reaching Stair-N. It can jump multiple stairs at a time, and the possible size of its jumps is given. For example, the figure below shows the different ways in which it can reach Stair-4 from the bottom (Stair-0) if it is allowed to jump 1 or 2 stairs at a time. Here the destination stair is N = 4, and the set of jump sizes is JS = {1, 2}.



Given the destination stair, N, and the set of jump sizes, JS, the frog wants to determine the jumps that will take it exactly to the destination stair. In some cases, a solution may not exist. For example, if N = 13 and JS = { 11, 7, 5 }, then the frog cannot reach Stair-N. The frog has heard that computers can help it to determine whether a solution exists, and if so, then what is the minimum number of jumps that takes it to the destination. In order to avoid too much complexity, it decides to confine itself to jumps of size a, b, and c only (that is, only three types of jumps). In order to help the frog, you are required to write the following recursive functions:

(a) Write the following recursive function in C to determine whether the destination, `dest,` can be reached using some combination of jumps from JS = {`a, b, c`}, where `a > b > c.`

```
int canreach(int dest, int a, int b, int c)
```

The function returns 1 if it is possible, and 0 otherwise. You are free to add more arguments in the function if needed.

[*Hint: N is reachable iff N=0 or at least one among* `N-a, N-b, N-c`, *is reachable*]

(b) Write the following recursive function in C to print a minimum length sequence of jumps that takes the frog from 0 to N.

```
int howtoreach(int dest, int a, int b, int c)
```

The function returns the number of jumps, or -1 if no solution exists. You are free to add more arguments in the function if needed.

[*Hint: The recursion is very similar to the previous one, but which jump will you try first?*]

(c) Write a program that reads the values of `dest, a, b, c,` and then reports the following. It uses the first function and reports whether the destination is reachable. If the answer to the first is affirmative, then it uses the second function to report the minimum number of jumps needed and the corresponding sequence of jumps. Please don't do everything with the second function only, even if that is possible.

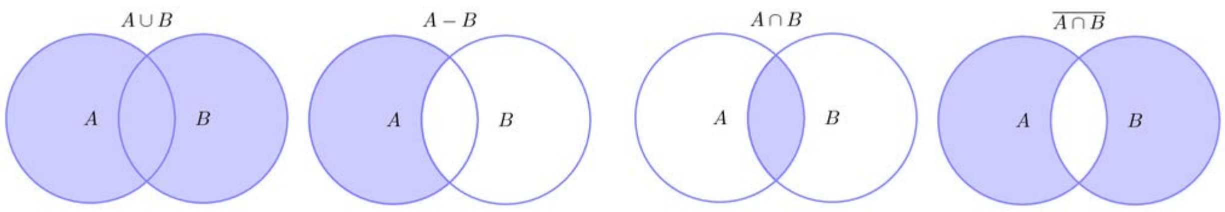
2. **Set Operations.** Arrays can be used to store sets of integers. This assignment seeks to implement the standard set operations on sets stored in arrays.

(a) Write the following functions:

```
int SetUnion( int a[], int na, int b[], int nb, int c[] )
int SetIntersect( int a[], int na, int b[], int nb, int c[] )
int SetMinus( int a[], int na, int b[], int nb, int c[] )
int SetSymDiff( int a[], int na, int b[], int nb, int c[] )
```

Each of these functions receive arrays `a[]` and `b[]` having `na` and `nb` elements respectively as input parameters. The array `c[]` stores the result of the set operation and the return value represents the number of elements in `c[].` The `SetUnion()` and `SetIntersect()` functions compute respectively the union and intersection of `a[]` and `b[]`. The `SetMinus()` function finds the elements that are in `a[]` but not in `b[]`. The `SetSymDiff()` function finds the symmetric difference between `a[]` and `b[],` that is those elements that are in one of the sets but not in both.

(b) Write a program that reads, na, `nb`, and the elements of sets A and B, and then uses the above four functions to find the elements in A∪B, A∩B, A–B and the symmetric difference of A and B. The program then prints the elements in the sets, A, B, A∪B, A∩B, A–B and the symmetric difference of A and B.


$A \cup B$
A
B
$A - B$
A
B
$A \cap B$
A
B
$\overline{A \cap B}$
A
B