Subject: PRF192- PFC
Workshop 06

Objectives: Managing arrays

Submission:
Please submit your work including a report and source code. All of them should be contained in a directory which is named as Workshop3_yourName_yourStudentID. Then zip this directory and submit.
The report MUST be a pdf file. Name of the file should contain your name and your student ID, such as Workshop3_yourName_yourStudentID.
The report must contain the pictures of all the test cases that you have done to test your programs.

Sample: Canadian SIN (Social Insurance Number)

SIN: 193 456 787 | check digit is 7 add first set of alternates to themselves
9 4 6 8 9 4 6 8 **18 8 12 16**
add the digits of each sum **1+8+8+1+2+1+6 = 27 (T1)**
add the other alternates 1+3+5+7 = 16 (T2)
total = T1+T2 = 27+16=43
Next highest integer multiple of 10 T3= 50 (50>43).
Difference T3-total = 50-43= 7 Matches the check digit, therefore this SIN is valid

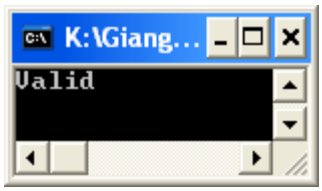
SIN: 193456787

N0	N1	N2	N3	N4	N5	N6	N7	N8	N9			
	1	9	3	4	5	6	7	8	7			
	9	4	6	8	9	4	6	8	18	8	12	16
C0	C1	C2	C3	C4	C5	C6	C7	C8	C9	C10	C11	C12

Algorithm for checking whether a number is a Canadian SIN or not

Use the array N, 10 elements, N[0] is not used
Use the array C, 12 elements, C[0] is not used
From n, computing N[i]:
From N, computing C[i]:
Compute
T1= C_9/10 + C_9%10 + C_{10}/10 + C_{10}%10 + C_{11}/10 + C_{11}%10 + C_{12}/10 + C_{12}%10
T2= N_1 + N_3 + N_5 + N_7;
Total= T1+T2
T3= (Total/10+1) *10; (Total=43 → T3 = (4+1)*10
If (T3-Total == N9) return “Valid”
else return “Invalid”

```c
#include <stdio.h>
/* Checking whether n is a Canadian SIN or not */
int checkCanadianSIN (int n)
{ int N[10]; /* array contains digits in n */
  int C[12]; /* array for checking */
  int T1, T2, T3, total; /* temporary values */
  int i, result=0; /* loop variable and result of the function */
  if (n>0)
  {  /* Compute N[i] */
     for (i=9; i>0; i--)
     {  N[i]= n%10;
        n= n/10;
     }
     /* Compute C[i] */
     C[1]=C[5]=N[2];C[2]=C[6]=N[4];C[3]=C[7]=N[6]; C[4]=C[8]=N[8];
     C[9]= 2*C[1]; C[10]=2*C[2]; C[11]=2*C[3]; C[12]=2*C[4];
     /* computer temporary values */
     T1= C[9]/10 + C[9]%10 + C[10]/10 + C[10]%10 +
         C[11]/10 + C[11]%10 + C[12]/10 + C[12]%10;
     T2= N[1] + N[3] + N[5] + N[7];
     total= T1+T2;
     T3=(total/10+1)*10;
     /* conclusion */
     if (T3-total == N[9])result=1;
   }
   return result;
}
int main()
{  int n= 193456787;
   /* n can be inputted */
   if (checkCanadianSIN(n)==1) puts("Valid");
   else puts("Invalid");
   getchar();
}
```



Refer to the sample above, write the following problem.

Problem 1 (4 marks)

An ISBN consists of exactly **10 digits**. The rightmost digit is the check digit. The check digit is validated modulo 11.

- multiply each digit from the first to the ninth by a weight from 10 to 2 respectively (the first digit by 10, the second by 9,..., the ninth by 2).
- the sum of the products plus the check digit should be divisible without remainder by 11.
- if there is a remainder, the whole number is not a valid ISBN.

Consider the following example:

ISBN 0003194876 | check digit is 6 add first set of alternates to themselves
0 0 0 3 1 9 4 8 7 10 9 8 7 6 5 4 3 2 **0 0 0 21 6 45 16 24 14** = 126
add check digit 6 total 132 divide by 11 12 remainder 0 Therefore this ISBN is valid.

I1	I2	I3	I4	I5	I6	I7	I8	I9	I10
0	**0**	**0**	**3**	**1**	**9**	**4**	**8**	**7**	**6**
C1	C2	C3	C4	C5	C6	C7	C8	C9	
0*10=0	**0*9=0**	**0*8=0**	**3*7=21**	**1*6=6**	**9*5=45**	**4*4=16**	**8*3=24**	**7*2=14**	

T= C1+ C2+C3+C4+C5+C6+C7+C8+C9 + I10; (T=132)
If (T%11==0) print out “Valid” else print out “Invalid”

Write a program that will accept a number (>=1 000 000 000) then show whether the number is an ISBN or not.

ISBN Validator ========== ISBN (0 to quit): 0003194876
This is a valid ISBN.
ISBN (0 to quit): 0003194875
This is not a valid ISBN. ISBN (0 to quit): 0
Have a Nice Day!

Problem 2 (6 marks)

Develop a C-program that helps user managing an 1-D array of real numbers(maximum of 100 elements) , with initial number of elements is 0, using the following simple menu:
1- Add a value
2- Search a value
3- Print out the array
4- Print out values in a range
5- Print out the array in ascending order
Others- Quit

- When the option 1 is selected, user will enters a value then it is added to the array
- When the option 2 is selected, user will enters a value then number of it’s existences will be printed out.
- When the option 3 is selected, values in the array will be printed out.
- When the option 4 is chosen, user will enter 2 values, minVal and maxVal, the values in array which are between minVal and maxVal are printed out (minVal <=value<=maxVal)
- When the option 5 is chosen, values in array will be printed out in ascending order but **their position are preserved. (sorting based their pointers only)**