Submission: Submit all your hand written contents as a pdf file (in Google Classroom).

Deadline: June 4 2020 11:00 pm [All the Answers must be submitted at once]

Problem 1. [10 Marks]
You are required to use the suggested data types for the Program 1. In this regard, you need to complete the gaps in Table 1. So, firstly fill in Column 2 (of Table 1) with the the digits of your Student ID (each digit should be in each row sequentially). Then get the Column 3 values from the corresponding information mentioned in Table 2. Table 2 contains the actual data type to be used for the individual digit of your Student ID.

DataType (mentioned)	Digit (in **S**tudent**ID**)	**DataType (to be used)**
datatype_1		
datatype_2		
datatype_3		
datatype_4		
datatype_5		
datatype_6		
datatype_7		
datatype_8		
datatype_9		
datatype_10		

Tabele 1: Data Types (to be replaced)

Digit (in StudentID)	**Data Type (actual)**	Size (in Byte)
0	char	1
1	short int	2
2	int	4
3	long int	8
4	float	4
5	double	8
6	long double	12
7	wchar_t	4
8	unsigned int	4
9	unsigned short int	2

Tabele 2: Data Types (to be used)

Program 1 (in C++) :

```
#include <iostream>
using namespace std;

struct struct1 {
    datatype_1 a;
    datatype_2 b;
    };
struct struct2 {
    datatype_3 c = 1;
    datatype_4 d = 2;
    struct sturct3 {
            datatype_5 e = 3;
            datatype_6 f = 4;
            } g;
    };


datatype_8 somefunc (datatype_8 p, datatype_9 q)
{
    struct2 r;
    datatype_8 s;

    if(q > 0)
        s = p * 2;
    else
        s = p * (-2);

    return s;
}

datatype_7 main ()
{
    datatype_8 i;
    datatype_9 j;
    datatype_10 k[3];
    struct1 m;

    i = 10;
    j = 20;

    somefunc(i, j);
    return 0;
}
```

As the answer of **Problem 1**, you need to redraw completed Table 1 & re-write Program 1 by replacing the datatype with the corresponding datatype considered in Table 1.

Problem 2. [20 Marks]
Create a symbol table structure (dynamic scoping) for the Program 1 (re-written in Problem 1) at line **27** [i.e. the line before returning to main function].

Problem 3. [20 Marks]
Suppose you are planning to create a Prolog database with the facts available from

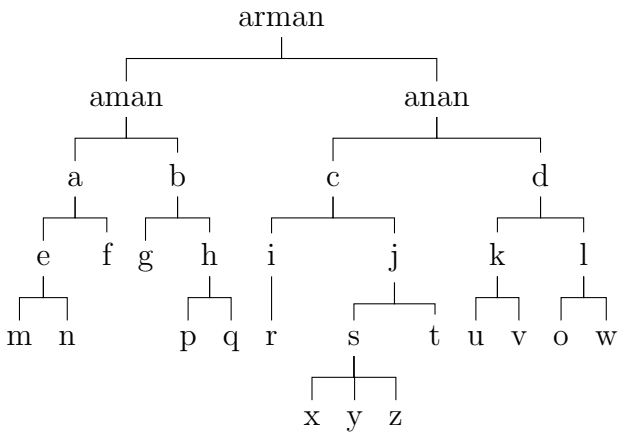


Figure 1: Family Tree

Figure 1. To represent a fact, for example - *arman is the father of aman*, in prolog you are planning to use the following strategy only:

father(arman , aman).

Similarly, there will be other facts - for example :

```
.....
father(arman , aman).
father(aman , anan).
father(aman , a).
father(aman , b).
father(anan , c).
father(anan , d).
father(a , e).
father(a , f).
father(b , g).
.....
.....
```

(a) Write down the completed database with all the facts for Figure 1.

The Prolog database is required to include two new rules. You need to define the rules for **relatives** and **cousins** (the defined rules consider both close and distant people):

(b) Write a rule for **relative(X,Y)** that succeeds if X and Y are relatives.

(c) Write a rule for **cousin(X,Y)** that succeeds if X and Y are cousins.

Now, you need to test and see the output of your defined rules with some queries. Each query will require two parameters. Let us consider these two parameters as α and β. In your query, α will be actually the first character of your name and β will be the second character of your name. For example, if your name is *asad* then your α will be 'a' and your β will be 's' (if you find α and β values similar then you will use the third character of your name as β and so on). You need to test your rules for **relatives** and **cousins** and show the changes in each corresponding variable used in the rule:

(d) Write down each step in Prolog search strategy for the query : **relative(**α**,**β**)**.

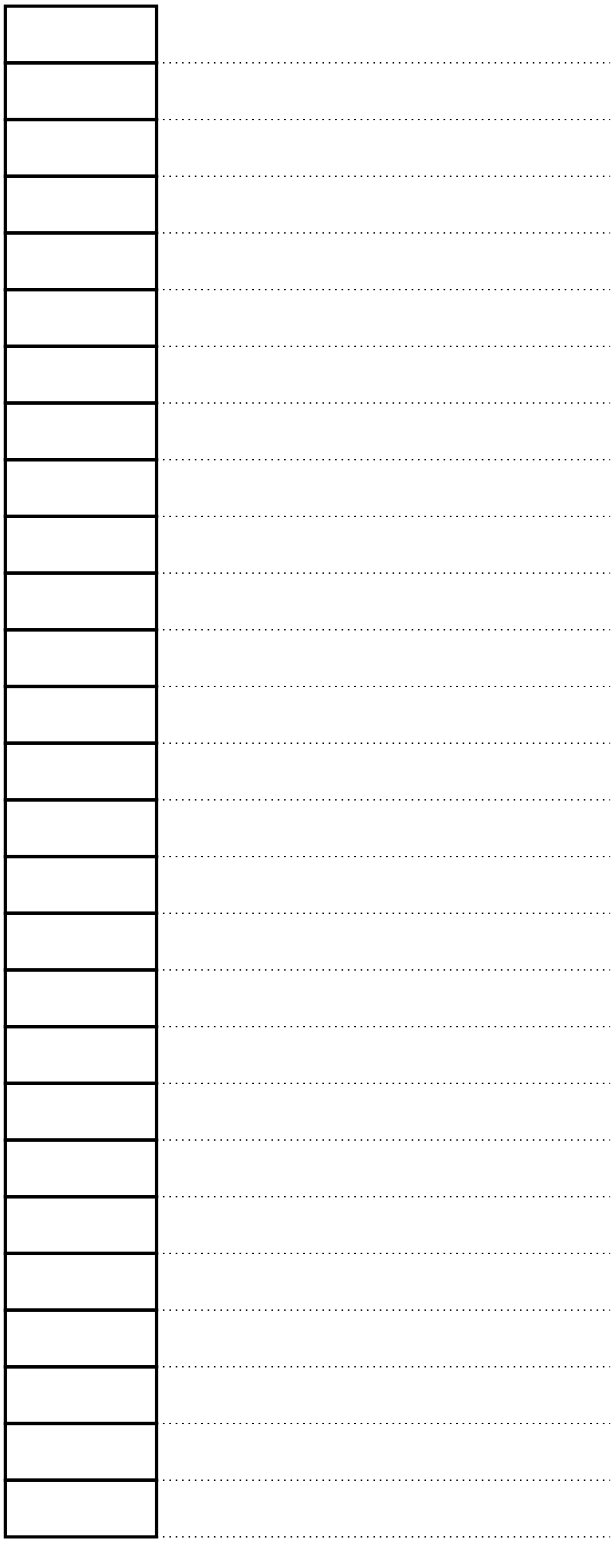
(e) Write down each step in Prolog search strategy for the query : **cousin(**α**,**β**)**.

Problem 4. [15 Marks]
Consider the grammar G = (V , T , S, P) where the start symbol is S with a set of non-terminals V = {S,A,B}, a set of terminals T = {c,f}, and set of productions P consisting of
$S \rightarrow cAB$
$A \rightarrow Bf$
$B \rightarrow \epsilon$

Create a parsing table for the grammar G and show the steps of a parsing program (changes in stack, input, output) while verifying the input ***cf*** with this grammar.

Problem 5. [35 Marks]
Show the **Activation** record (draw the stack contents) for the Program 1 (re-written in Problem 1) while the execution started from the main function and the execution reached at line **27** [i.e. the line before returning to main function]. Again, write down the corresponding codes generated for the main function and somefunc function that will be loaded in the code segment of the memory. (Your answer should be in the format shown in next page; you may print that page to answer this question).

Stack type size offset



somefunc:

main: