German University in Cairo
Media Engineering and Technology
Dr. Nada Sharaf

July 10, 2021

Concepts of Programming Languages
Spring 2021
Final Exam

Bar Code

Instructions: Read carefully before proceeding.

1) Duration of the exam: 3 hours.
2) (Non-programmable) Calculators are allowed.
3) No books or other aids are permitted for this test.
4) This exam booklet contains 13 pages, including this one. Three extra sheets of scratch paper are attached and have to be kept attached. **Note that if one or more pages are missing, you will lose their points. Thus, you must check that your exam booklet is complete**.
5) Write your solutions in the space provided. If you need more space, write on the back of the sheet containing the problem or on the four extra sheets and make an arrow indicating that. **Scratch sheets will not be graded unless an arrow on the problem page indicates that the solution extends to the scratch sheets**.
6) When you are told that time is up, stop working on the test.

Good Luck!

Don't write anything below `;-)`

Exercise	1	2	3	4	5	6	$\sum$
Possible Marks	10	8	20	6	12	18	74
Final Marks							

Exercise 1

Use clpfd to solve this puzzle. We have a 4*4 grid of positive numbers. Numbers cannot exceed the value 100. Values have to be distinct. The first value of the first row has to be 1. The last value of the last row has to be 100. Each row has to have incrementing numbers i.e. if the first cell in the row is 10, the rest of the row has to have values>10. In addition, all the values in a row have to be greater than the values in all previous rows.

The below is an example of a solution for this puzzle:

1	5	9	20
35	36	40	42
50	60	70	80
90	95	98	100

Use clpfd to model this problem.

```
:-use_module(library(clpfd)).
```

Exercise 2 Prolog

Implement a Prolog predicate `my_split/3`. `my_split(S,L,O)` is true only if `O` is a list of lists representing the result of splitting the list by `S`.

```
Examples:

?- my_split(a,[b,c,a,d,e,a,b],O).
O = [[b, c], [d, e], [b]] ;
false.

?- my_split(a,[b,c,a,d,e,a,b,a],O).
O = [[b, c], [d, e], [b]] ;
false.

?- my_split(a,[a,b,c,a,d,e,a,b,c],O).
O = [[], [b, c], [d, e], [b, c]] ;
false.
```

Solution:

```
my_split(_,[],[]).
my_split(S,[S|T],[[]|O]):- my_split(S,T,O).
my_split(S,[H|T],[[H]]):-
                    H\=S,
                    my_split(S,T,[]).
my_split(S,[H|T],[[H|HS]|TS]):-
                    H\=S,
                    my_split(S,T,[HS|TS]).
```

Exercise 3 Haskell

Given the below data type definition

```
data Students = Student Int String String deriving (Show, Eq)
data Courses = Course Int String deriving (Show, Eq)

type Enrollment = [(Students, Courses, Int)]
```

Each student is defined through their ID, name and major. Each course is defined through its ID and name. We keep track of data representing which student is enrolled in which course along with the grade they got.

Answer the following. **You can use any predefined functions needed**

a) Given the below mystery function:

```
mystery course enrollment =
 filter (\(Student id name major, Course cid cname, grade) -> cname==course) enrollment
```

1. What is the functionality of `mystery`
2. What is the type of `mystery`

You are allowed to use the mystery function in the rest of the question

b) Implement a function `groupCourses` that takes as an input a list of course names and a an enrollment list. The function should return a list that contains a pair for every course containing its name and a list of the enrolled students

```
Examples:
> groupCourses ["concepts","game"] [(Student 1 "Nada" "MET", Course 403 "concepts", 98),
(Student 1 "Rana" "MET", Course 403 "concepts", 80), (Student 1 "Rana" "MET",
Course 401 "game", 90),(Student 1 "Ahmed" "Mecha", Course 401 "game", 85),
(Student 1 "Rami" "Mecha", Course 401 "game", 80)]

[("concepts",[(Student 1 "Nada" "MET", Course 403 "concepts",98),
(Student 1 "Rana" "MET",Course 403 "concepts",80)]), ("game",[(Student 1 "Rana" "MET",
Course 401 "game",90), (Student 1 "Ahmed" "Mecha", Course 401 "game",85),
(Student 1 "Rami" "Mecha", Course 401 "game",80)])]

> groupCourses ["concepts","game","web"] [(Student 1 "Nada" "MET",
Course 403 "concepts", 98), (Student 1 "Rana" "MET", Course 403 "concepts", 80),
(Student 1 "Rana" "MET", Course 401 "game", 90)]

[("concepts",[(Student 1 "Nada" "MET",Course 403 "concepts",98),
(Student 1 "Rana" "MET",Course 403 "concepts",80)]),
("game",[(Student 1 "Rana" "MET",Course 401 "game",90)]),("web",[])]
```

Solution:

```
groupCourses [] enrollment = []
groupCourses ( cname:restCourses) enrollment
 = (cname,mystery cname enrollment) : groupCourses restCourses enrollment
```

c) Implement a function `maxLen` that takes as an input a list of pairs representing the course name and the students enrolled in the course. The function returns the pair representing the course with the maximum number of enrolled students

```
maxLen [("concepts",[(Student 1 "Nada" "MET",Course 403 "concepts",98),
(Student 1 "Rana" "MET",Course 403 "concepts",80)]),
("game",[(Student 1 "Rana" "MET",Course 401 "game",90),
(Student 1 "Ahmed" "Mecha",Course 401 "game",85),(Student 1 "Rami" "Mecha",
Course 401 "game",80)])]

("game",[(Student 1 "Rana" "MET",Course 401 "game",90),
(Student 1 "Ahmed" "Mecha",Course 401 "game",85),
(Student 1 "Rami" "Mecha",Course 401 "game",80)])
```

Solution:

```
maxLen [(s,l)] = (s,l)
maxLen ((s,l):xs) = if length l > length tl then (s,l) else (ts,tl) where (ts,tl
```

Exercise 4 Higher Order Functions

A build-in higher ordered function is `foldl`. The function also folds a list similar to hat `foldr` does. However, both functions use different orders.

For example `foldr (-) 20 [10,4,3]` is computed as follows 10-(4-(3-20)) producing `-11`

However, `foldl (-) 20 [10,4,3]` is computed as follows ((20-10)- 4)-3 producing `3`

a) Implement the function `foldl`

   Solution:

```
foldl2 f b [] = b
foldl2 f b [x] = f b x
foldl2 f b (x:xs) = foldl2 f  (f b x) xs
```

b) **Bonus** Using foldr, implement foldl without recursion. You can implement extra helper non-recursive functions.

Exercise 5

In this question, you are allowed to use any needed pre-defined functions

a) Implement a function `part l n` that returns a pair (p1,p2) such that p1 is a list with the first n items of list l and p2 contains the rest of the elements of list l

```
Examples:
> part [10,6,2,3,4,15,20] 3
([10,6,2],[3,4,15,20])


> part [10,6,2,3,4,15,20] 10
([10,6,2,3,4,15,20],[])

> part [] 2
([],[])
```

Solution:

```
 parth [] _ _ = ([],[])
parth (x:xs) n m |n<=m = (x: rest, recrest)
|otherwise = ([],x:xs)
where (rest, recrest) = parth xs (n+1) m
```

b) Implement a function `swapevery l n` that swaps some elements of the list `l` as follows. We swap the start and end of every n-element sub-list. For example `swapevery [1,2,3,5,6,8] 3` will swap first the 1 and the 3 from the first sub-list [1,2,3] then 5 and 8 are swapped. Thus, the new list is `[3,2,1,8,6,5]`

```
> swapevery [1,2,3,4,5,6,7,8] 4
[4,2,3,1,8,6,7,5]

> swapevery [1,2,3,4,5,6,7,8] 3
[3,2,1,6,5,4,7,8]

> swapevery [1,2,3,4,5,6,7,8] 2
[2,1,4,3,6,5,8,7]
```

Exercise 6

Trace each of the following:

a)
```
#include <stdio.h>

void f(int v, int *pv, int **ppv)
{
  int v2, v3;
  *(*ppv) = *(pv)+3;
  printf( "%d \n", **ppv);
  *pv = *pv +  2;
  printf( "%d \n", *pv);
   v = v + 3;
   printf( "%d \n", *pv);
   printf( "%d \n", **ppv);

}

int main()
{
   int x, *y, **z;
   x = 4;
   y = &x;
   z = &y;
  f(x,y,z);
   return 0;
}
```

b)
```
int main()
{
  int array[10] = { 50,10,20,1,2,8,10,8,9,0};

  int *ptr = &array[5];
  for(int i=0;i<5;i++)
  {
    *(ptr+i) = (i+5);
  }

  for(int i=0;i<10;i++)
  {
     printf( "%d \n", array[i]);
  }
   return 0;
}
```

Scratch paper

Scratch paper

Scratch paper