BBM301 FINAL EXAM

YILDIRIM BAYAZIT AKYÜREK

TOTAL POINTS

59 / 100

QUESTION 1

1 Q1 **9 / 16**

- **0 pts** All correct
- **3 pts** a wrong
- **4 pts** b wrong
- ✓ **3 pts** **c wrong**
- **3 pts** d wrong
- **3 pts** e wrong
- ✓ **2 pts** **t and f written wrong in b and e**
- **1 pts** t written wrong in e / t written wrong in b
- ✓ **2 pts** **a should be list**
- **1 pts** minor error in a
- **16 pts** Empty or wrong answer

QUESTION 2

2 Q2 **3 / 12**

- ✓ **+ 4 pts** **one answer correct**
- **+ 8 pts** two answers correct
- **+ 12 pts** All Correct: **'(2)** and **append** and *** 2 n**
- **+ 0 pts** Blank/ Incorrect
- ✓ **1 pts** **a minor mistake**

QUESTION 3

3 Q3 **5 / 12**

- **+ 6 pts** (a) Correct: **DateName**
- **+ 6 pts** (b) Correct: **IDDate**
- **3 pts** Minor mistake in (a)
- **3 pts** Minor mistake in (b)
- ✓ **+ 5 pts** **Partially correct**
- **+ 0 pts** Blank

QUESTION 4

4 Q4 **21 / 24**

- ✓ **+ 1 pts** **a) a = 7**
- ✓ **+ 1 pts** **a) b = 1**
- ✓ **+ 1 pts** **a) c = 3**
- ✓ **+ 1 pts** **a) d = 9**
- ✓ **+ 1 pts** **a) e = 10**
- **+ 0 pts** a - not answered / all wrong
- ✓ **+ 1 pts** **b) a chain 1**
- ✓ **+ 1 pts** **b) a local 3**
- ✓ **+ 1 pts** **b) c chain 2**
- ✓ **+ 1 pts** **b) c local 4**
- ✓ **+ 1 pts** **b) e chain 0**
- ✓ **+ 1 pts** **b) e local 3**
- **+ 0 pts** b - not answered / all wrong
- ✓ **+ 1 pts** **c) a = 4**
- ✓ **+ 1 pts** **c) b = 8**
- ✓ **+ 1 pts** **c) c = 6**
- ✓ **+ 1 pts** **c) d = 10**
- ✓ **+ 1 pts** **c) e = 10**
- **+ 0 pts** c) not answered / all wrong
- **+ 2 pts** d) a = sub1, sub3
- ✓ **+ 2 pts** **d) b = bigsub, sub4**
- ✓ **+ 2 pts** **d) c = bigsub, sub3**
- **+ 2 pts** d) d = sub1, sub4
- **+ 0 pts** d) not answered / all wrong
- **+ 4 pts** d) Values are given/ not the subprogram names
- **+ 2 pts** Only top of the stack is shown
- **+ 1 Point adjustment**

top of the stack is shown for a and d

QUESTION 5

5 Q5 **8 / 16**

- ✓ **+ 6 pts** **a - Correct**
- **+ 3 pts** Only 1 parse tree is correct
- **+ 0 pts** a - wrong
- **+ 0 pts** a - not answered

+ **1 pts** start -> dong

+ **1 pts** ding -> terminal DING __ | __

+ **1 pts** ding -> __ DING ding | __

✓ **+ 1 pts** **ding -> __ DING __ | terminal**

+ **1 pts** dong -> dong DONG __ | __

✓ **+ 1 pts** **dong -> __ DONG dell | __**

+ **1 pts** dong -> __ DONG __ | dell

+ **1 pts** dell -> ding DELL __ | __

+ **1 pts** dell -> __ DELL dell | __

+ **1 pts** dell -> __ DELL __ | ding

+ **0 pts** b - wrong

+ **0 pts** b - not answered

+ **0 pts** a - two different strings

QUESTION 6

6 Q6 13 / 20

- **0 pts** Correct

- **6 pts** a is wrong

✓ **- 7 pts** **b is wrong**

- **7 pts** c is wrong

- **20 pts** All wrong or empty

BBM 301 - Programming Languages - Fall 2021 Final Exam
January 07, 2022

Name: Y. Bayezit Akkaleş

Student ID number: 21504343

1- [16pts] Scheme. For each of the following Scheme expressions, please show the result. If the expression results in a run-time error, write error.

```
(define lis1 '(2 3 4))
```

- `(cons (+ 1 2) '(5 5))` output: 3 5 5
- `(map (lambda (x) (>= x 3)) '(7 5 1))` output: T T F
- `(* 10 (car '((2) (4) (6))))` output: 20
- `(eval (cons '* lis1) )` output: 24
- `(odd? (car (map (lambda (x) (* 5 x)) (cdr lis1))))` output: T

15 20 3.4

2- [12pts] Scheme. In the following Scheme code, the aim is the construct the list (2 4 6 ... 2N) recursively for an input N. For example, for N=5, the output should be (2 4 6 8 10). Please fill in the blanks to complete the code.

```
(define (Sequence N)
   (if (= N 1) 2
       (* N (Sequence (- N 1)))
       (list (* 2 N))))))
```

3- [12pts] Lex. Consider the following lex file:

```
binary [01]
decimal [0-9]
hexadigit [0-9A-F]
alphabetic [a-zA-Z]
%%
{alphabetic}{decimal}+        printf("Date");
{hexadigit}*                  printf("ID");
({binary} | {alphabetic})+    printf("Name");
%%
```

a) What is the output for the following input: **A09ZG10**

Name Date

b) What is the output for the input: **70112DF0310z0**

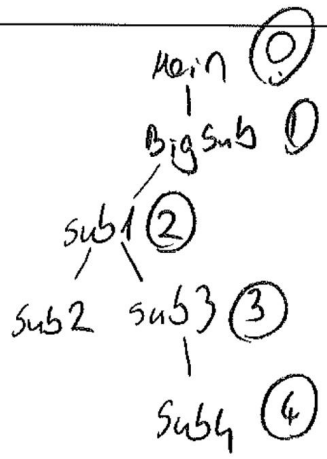
ID Name ID Date

NAME: Y. Bayazıt Alıştırma ID: 2180343

4. [24pts] Subprogram Implementation and Scoping

Given the following program, answer the questions below.

```
program MAIN
  procedure BigSub
  int b = 1, c = 3;
    procedure sub1
    int a = 7, d = 9;
      procedure sub2
      int e;
      begin {sub2}
        e = a + c ;
        print(a, b, c, d, e);
      end {sub2}
      procedure sub3
      int a = 4, c = 6;
        procedure sub4
        int b = 8, d = 10;
        begin {sub4}
          sub2;
        end   {sub4}
      begin {sub3}
        sub4;
      end   {sub3}
    begin {sub1}
      sub3;
    end   {sub1}
  begin{BigSub}
    sub1;
  end{BigSub}
begin {MAIN}
  BigSub;
end   {MAIN}
```



a) (5 pts) Assuming that static scoping is used, what will be the output of the program?

7, 1, 3, 9, 10

b) (6 pts) Give the **(chain_offset, local_offset)** pairs at **point *** for the following variables

	chain_offset	local_offset
a	3-2 = 1	3
c	3-1 = 2	4
e	0	3

c) (5 pts) Assuming that dynamic scoping is used, what is the output of the program? **Note that the calling sequence is MAIN->BigSub->sub1->sub3->sub4->sub2**

b=1 a=7 a=4 b=8 e=4+6=10
c=3 d=9 c=6 d=10

4, 8, 6, 10, 10

d) (8 pts) Assume that dynamic scoping is implemented using the shallow-access method with a stack for each variable name. Show the contents of stacks (subprogram names) associated with the variable names {a,b,c,d} at the time of the execution of sub2 (when execution reaches to *).

	sub4	sub3	
sub3	bigsub	bigsub	sub4
a	**b**	**c**	**d**

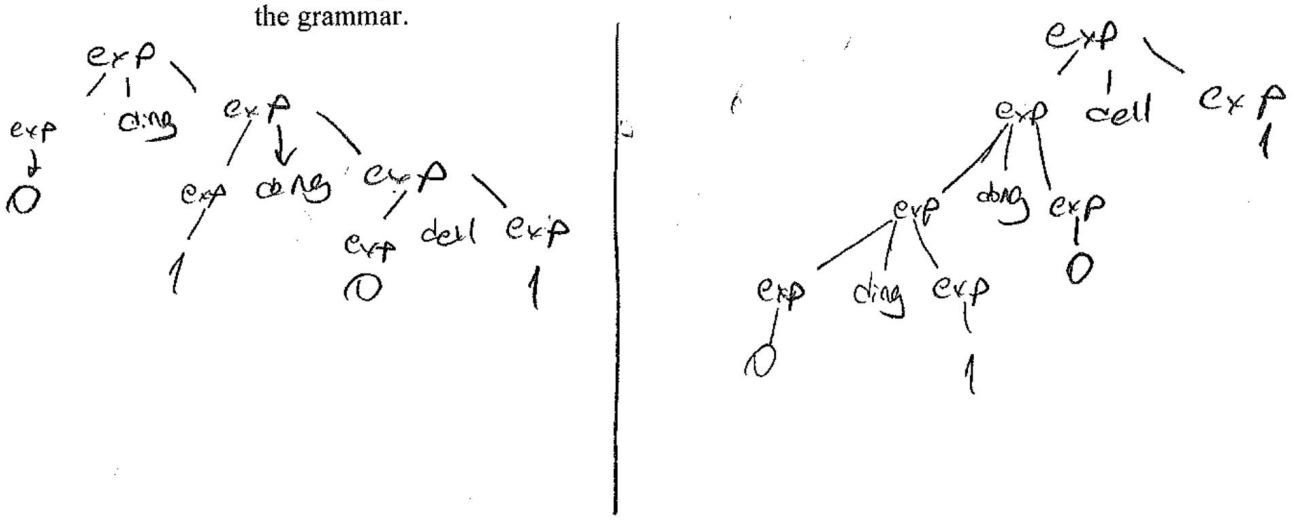
NAME: Y. Bayzat Akyüz ID: 21904343

5- [16 pts] BNF and Ambiguity:

a) (6 pts) Consider the following grammar

```
<exp> -> <exp> DING <exp>
       |  <exp> DONG  <exp>
       |  <exp> DELL <exp>
       | 0
       | 1
```

Draw two parse trees for the string " 0 DING 1 DONG 0 DELL 1" to prove the ambiguity of the grammar.



b) (10 pts) Now, complete the following grammar to resolve the ambiguity of the grammar above according to the rules below:

- DING has the highest precedence followed by DELL, and DONG has the lowest precedence.
- DELL and DING are right associative, and DONG is left associative

Note that, DING, DONG, DELL are the operators, and `<start>`, `<ding>`, `<dong>`, `<dell>` and `<terminal>` are the non-terminals. The new grammar will start with `<start>`

```
<start> -> <ding>
<ding>  -> <dong> DING <dell>  | terminal
<dong>  -> <ding> DONG <dell>  | terminal
<dell>  -> dong   DELL ding    | terminal
<terminal> -> 0 | 1
```

NAME: Yildirim Bayazit Akyüzer ID: 21806343

6. [20pts] Parameter Passing

Consider the following program. Please answer the following questions assuming that the language is **a statically scoped** language, where **the array indexes start with 1**.

```
int list[4]={3,5,4,2};
int i;

void fun(int a, int b) {
   i = 2;
   if (a < b){
      a = a - i ;
      i = i + 1 ;
   else
      b = b + 2 ;
   }
}

int main() {
   i=1;
   fun(list[i],list[i+1]);
   printf("%d %d %d %d\n", list[1],list[2],list[3],list[4]);
   return;                                                    0;
}
```

What will be the output of the program if the following parameter passing methods are used:

(a) *pass-by-reference*

1, 5, 4, 2

(b) *pass-by-value-result* (assume that **the address is taken at the return**)

1, 5, 4, 2

(c) *pass-by-name*

3, 5, 6, 2