име :	фн	1/15

СОФИЙСКИ УНИВЕРСИТЕТ “СВ. КЛИМЕНТ ОХРИДСКИ” ФАКУЛТЕТ ПО МАТЕМАТИКА И ИНФОРМАТИКА
ДЪРЖАВЕН ИЗПИТ ЗА БАКАЛАВЪРСКА СТЕПЕН ПО ИНФОРМАТИКА – 29-30.04.2003 г.

Оценки върху практическите задачи: а) **b)** **общо:**
Забележки:

Условие на задача А :

Дадена е автоматната граматика Г = < N, T, S, P > , където:

N = { **S**, **A**, **B**, **C**, **D** }
T = { 0, 1 }
S ε **N**
P = { **S** → 0**S**|1**S**|0**A**|1**B**, **A** → 0**A**|1**A**|1**C**, **B** → 0**B**|1**B**|0**D**, **C** → 0, **D** → 1 }

Постройте краен детерминиран автомат, който разпознава езика, породен от граматиката Г.

Решение на задачата :

1. Каква ще бъде стойността на x след изпълнението на следния оператор:

```
int  x = - 3 * 4 % - 6 / 5;                    //      x =
```

2. Каква ще бъде стойността на x след изпълнението на следния фрагмент от програма:

```
int  x = 2;
x *= 3+2;                                      //      x =
```

3. Каква ще бъде стойността на x и z след изпълнението на следния фрагмент от програма:

```
int x, y, z;
x=y=1;
z = x++ - 1;
z+= - x++ + ++y;                               //      x =          z =
```

4. Кой от операторите за цикли, означени с буквите от A до D, е еквивалентен на следния:

```
while ( U )   { if ( V ) continue;
                W; }
```

A) `while ( U ) { W; if ( V ) break;}`
B) `while ( U ) { W; if ( !V ) break;}`
C) `while ( U ) if ( !V ) W;`
D) `while ( U ) { if ( V ) continue;`
`else break;`
`W; }`

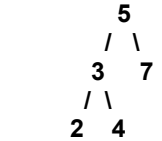
5. Колко въпроса са необходими за оптималната стратегия за познаване на едно число между 1 и 1000, ако се разчита на верните отговорине на въпроси, на които се отговаря само с “да/не”?

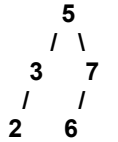
A) 1000 B) 999 C) 500 D) 32 E) 10

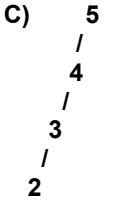
6. Кое от следните десетични числа винаги има точно представяне в двоична бройна система?

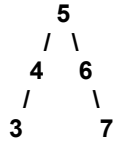
A) 0.1 B) 0.2 C) 0.3 D) 0.4 E) 0.5

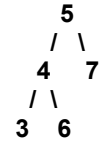
7. Кое от следващите дървета не е двоично дърво за претърсване?

A)


B)


C)


D)


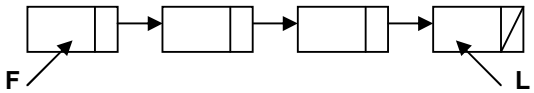
E)


7. Какви ще са стойностите на i и j след изпълнението на следния фрагмент?

```
void p(int *k, int  *m)
   {* k = *k - *m; *m = *k + *m; *k = *m - *k;}
void main()
   {int I = 2, j = 3;
   p(&i,&j);
   ……..
```

A) i=0, j=2 B) i=1, j=5 C) i=2, j=3 D) i=3, j=2 E) нито едно от изброените

9. Да разгледаме свързан списък от вида:



където F е указател към първия елемент на списъка, а L е указател към последния елемент на списъка. Времето на коя от следните операции зависи от дължината на списъка:

A) Изтриване на последния елемент на списъка;
B) Изтриване на първия елемент на списъка;
C) Добавяне на елемент след последния елемент на списъка;
D) Добавяне на елемент преди първия елемент на списъка;
E) Разменяне на първите два елемента на списъка.

10. Колко реда ще отпечата следната програма?

```
#include <stdio.h>
void main()
{  float sum = 0.0, j=1.0, i=2.0;
   while (i/j > 0.001)
     { j=j+j;
       um = sum + i/j;
       printf("%f\n",sum);
     }
}
```

A) 0-9 B) 10-19 C) 20-29 D) 30-39 E) Повече от 39

11. Кое от следните цели числа е най-близко до последното отпечатано число в горната програма:

A) 0 B) 1 C) 2 D) 3 E) 4

12. Цялото число c е делител на двете цели числа x и y тогава и само тогава, когато c е делител на x и c е делител на y.
Кое от следните множества от цели числа е множеството от всички общи делители на две цели числа:

A) {-6, -2, -1, 1, 2, 6}
B) {-6, -2, -1, 0, 1, 2, 6}
C) {-6, -3, -2, -1, 1, 2, 3, 6}
D) {-6, -3, -2, -1, 0, 1, 2, 3, 6}
E) {-6, -4, -3, -2, -1, 1, 2, 3, 4, 6}

13. Кой от следните префиксни изрази е еквивалентен на израза ((2 + 3) * 4 + 5 *(6 + 7) * 8) + 9

A) + + * + 2 3 4 * * 5 + 6 7 8 9
B) * + + 2 3 4 * * 5 + 6 7 8 9
C) * + + 2 3 4 * * 5 + + 6 7 8 9
D) * + + + 2 3 4 * * 5 + 6 7 8 9
E) + * + * 2 3 4 + + 5 * 6 7 8 9

14. Нека P е рекурсивна функция, т.е. в тялото ѝ има обръщение към самата нея.
Кои от следните изисквания гарантират, че работата на P винаги ще завършва:

I. P има поне една локална променлива.
II. Съществува клон от изпълнението на P, в който няма обръщение към P.
III. P използва глобална променлива или има поне един параметър.

A) Само I
B) Само II
C) Само I и II
D) Само II и III
E) I, II и III

15. Нека S е операторът for (i=1;i<=N;i++) V[i]=V[i]+1;
Кои от следните фрагменти променят V както S:

```
 I. i=0;
    while(i<=N) {i=i+1;V[i]=V[i]+1;}
II. i=1;
    while(i<N) {V[i]=V[i]+1;i=i+1;}
III. i=0;
    while(i<N) {V[i+1]=V[i+1]+1;i=i+1;}
```

A) Само I
B) Само II
C) СамоIII
D) Само II и III
E) I, II и III

16. Двойносвързан списък е дефиниран като:

```
struct Element
{int Value;
 struct Element *Fwd, *Bwd;};
```

където Fwd и Bwd са указатели към следващия и предишния елемент в списъка.
Кой от следните фрагменти може да изтрива елемента, указван от X от двойносвързания списък, ако той не е нито първият, нито последният елемент в списъка:

```
A) X->Bwd->Fwd = X->Fwd ;
   X->Fwd->Bwd = X->Bwd ;
B) X->Bwd->Fwd = X->Bwd ;
   X->Fwd->Bwd = X->Fwd ;
C) X->Bwd->Bwd = X->Fwd ;
   X->Fwd->Fwd = X->Bwd ;
D) ->Bwd->Bwd  = X->Bwd ;
   X->Fwd->Fwd = X->Fwd ;
E) X->Bwd = X->Fwd ;
   X->Fwd = X->Bwd ;
```

17. Структура от данни се състои от възли, всеки от които има точно два указателя към други възли и указателите не са нулеви. Следната програма на C трябва да бъде използвана за да намери броя на възлите, достижими от даден възел. Тя използва поле с име mark, което е инициализирано с нула за всички възли. Само един оператор липсва в текста.

```
struct test {int info, mark; struct test *p, *q;}
int nodecount (struct test *a)
{ if (a->mark) return 0;
  return nodecount(a->p) + nodecount(a->q) + 1;}
```

Какво трябва да се промени, за да може програмата да заработи правилно:

A) Да се добави " a->mark = 1 ;" като първи оператор;
B) Да се добави " a->mark = 1 ;" след оператора if;
C) Да се добави " a->mark = 1 ;" като последен оператор;
D) Да се добави " a->mark = 0 ;" след оператора if;
E) Да се добави " a->mark = 0 ;" като последен оператор.

18. Да разгледаме следния програмен фрагмент:

```
int f(int x)
{if (x<1) return 1;
  else return f(x-1) + g(x);}
int g(int x)
{if (x<2) return 1;
  else return f(x-1) + g(x/2);}
```

Кое от следните най-добре определя нарастването на f(x) като функция на x :

A) Логаритмично B) Линейно C) Квадратично D) Кубично E) Експоненциално

19. Открийте, обяснете и коригирайте грешките в следните програмни фрагменти:

a)
```
class Example{
public:
  Example(int y=10)
  {data = y;}
  int getIncrementData() const
  { return ++data;}
  static int getCount()
  { cout << "Data=" << data << endl;
    return count;}
private:
  int data;
  static int count;
}
```

b)
```
  char *string;
  string = new char[20];
  free(string);
```

20. Какво прави (извежда) тази програма?

```
#include <iostream.h>
int mystery2(const char *s)
{ for (int x=0; *s!='\0'; s++)
      ++x;
  return x;
}
void main()
{ char string[80];
  cout << "Въведете низ: ";
  cin >> string;
  cout << mystery2(string) << endl;
}
```

21. Какво прави (извежда) тази програма?

```
#include <iostream.h>
void mystery1(char *s1, const char *s2)
{ while (*s1 != '\0')
       ++s1;
  for( ; *s1=*s2; s1++, s2++);
}
void main()
{ char string1[80], string2[80];
  cout << "Въведете два низа: ";
  cin >> string1 >> string2;
  mystery1(string1, string2);
  cout << string1 << endl;
}
```

22. Открийте, обяснете и коригирайте (ако е възможно) грешките в следните програмни фрагменти::

a)
```
  int *number;
  cout << number << endl;
```

b)
```
  float *realPtr;
  long *integerPtr;
  integerPtr = realPtr;
```

c)
```
  int *x, y;
  x = y;
```

d)
```
  char s[] = "Това е масив от символи";
  for ( ; *s != '\0'; s++)
  cout << *s << ' ';
```

e)
```
  int *numPtr, result;
  void *genericPtr = numPtr;
  result = *genericPtr + 7;
```

f)
```
  float x = 19.34;
  float xPtr = &x;
  cout << xPtr << endl;
```

g)
```
  char *s;
  cout << s << endl;
```

23. Какво прави (извежда) тази програма?

```
#include <iostream.h>
void mystery3(const char *s1, const char *s2)
{ for( ; *s1 != '\0' && *s2 != '\0'; s1++, s2++)
      if (*s1 != *s2)
           return 0;
  return 1;
}
void main()
{ char string1[80], string2[80];
  cout << "Въведете два низа: ";
  cin >> string1 >> string2;
  cout << "Резултатът = " << mystery3(string1, string2) << endl;
}
```

24. Какво прави (извежда) тази програма?

```
#include <iostream.h>
class CreateAndDestroy{
public:
  CreateAndDestroy(int);
  ~CreateAndDestroy();
private:
  int data;
};
CreateAndDestroy::CreateAndDestroy(int value)
{ data = value;
  cout << "Обект " << data << " конструктор";
}
CreateAndDestroy::~CreateAndDestroy()
{ cout << "Обект " << data << " деструктор";}
void create();

CreateAndDestroy first(1);
void main()
{ cout << " (външен създаден до main)" << endl;
  CreateAndDestroy second(2);
  cout << " (вътрешен автоматичен в main)" << endl;
  static CreateAndDestroy third(3);
  cout << " (вътрешен статичен в main)" << endl;
  create();
  CreateAndDestroy fourth(4);
  cout << " (вътрешен автоматичен в main)" << endl;
}
void create()
{ CreateAndDestroy fifth(5);
  cout << " (вътрешен автоматичен в create)" << endl;
  static CreateAndDestroy sixth(6);
  cout << "(вътрешен статичен в create)" << endl;
  CreateAndDestroy seventh(7);
  cout << "(вътрешен автоматичен в create)" << endl;
}
```

25. Открийте, обяснете и коригирайте грешките в следния програмен фрагмент:

a)
```
class Array{
public:
  . . .
  const Array &operator=(const Array &);
  int operator!=(const Array &)const;
  int operator[](int);
private:
  int *ptr; //Указател към първия елемент на масива
  int size; //Брой на елементите на масива
}
```

b)
```
const Array &Array::operator=(const Array &right)
{ delete[]ptr;
  size = right.size;
  ptr = new int[size];
  assert(ptr!=0);
  for(int i=0; i<size; i++)
    ptr[i]=right.ptr[i]
  return *this;
}
```

c)
```
int Array::operator!=(const Array &right)const
    { for(int i=0; i<size; i++)
        if (ptr[i]!=right.ptr[i])
           return 1;
      return 0;
    }
```

d)
```
int Array::operator[](int subscript)
{ assert(0<=subscript && subscript<size);
  return ptr[subscript];
}
```

26. Да се открие грешката, да се обясни и коригира в следния клас Increment:

```
class Increment{
public:
  Increment( int c=0, int i=1);
  void addIncrement() {count+=increment;}
private:
  int count;
  const int increment;
}

Increment::Increment(int c, int i)
{ count = c;
  increment = i;
}
```

27. Функцията *push* добавя елемент, а функцията *pop* изключва елемент от последователен стек. Многоточето да се замени с липсващия оператор в следните фрагменти:

```
#define M 100
struct stack
{ int t; // номер на елемента във върха на стека
  float stack_array[M];
};
```

a)
```
void push(const stack *s, float x)
{ . . .
  if(s->t > M)
     { cout << "\nПРЕПЪЛВАНЕ\n";
       exit(1); }
  s->stack_array[s->t - 1] = x;
}
```

b)
```
float pop(const stack *s)
{ . . .
  s->t--;
  return s->stack_array[s->t];
}
```

28. Функцията push добавя елемент в свързан стек, а главната функция я използва. Многоточето да се замени с липсващия оператор в следните фрагменти:

```
struct stack_el
{ int info;
  stack_el *link;
};
```

a)
```
void push(stack_el **t, int x)
{ stack_el *p;
  if ((p=new stack_el) == NULL)
    { cout << "\nНяма свободна памет\n";
      exit(1);
    }
  p->info = x;
  . . .
  *t = p;
}
```

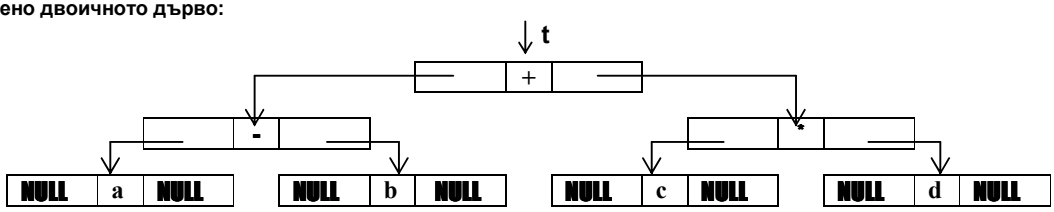
b)
```
void main()
{ . . .
  push(&stack, 1);
  push(&stack, 2);
}
```

29. Функцията dequeue изключва елемент от свързана опашка с първи фиктивен елемент, който се сочи от f.

```
struct queue_el
{ float info;
  queue_el *link;
};
void dequeue(queue_el *f, queue_el *r)
{ float x;
  queue_el *p = f->link;
  if (p == NULL)
    { cout << "\nОпашката е празна\n";
      exit(1); }
  . . .
  x = p->info;
  delete p;
  if (f->link == NULL)
     r = f;
  return x;
}
void main()
{ float y;
  queue_el Q, *F = &Q, *R;
  y = dequeue(F, &R);
}
```

a) Многоточето във функцията dequeue да се замени с липсващия оператор.
b) Да се открият, обяснят и коригират грешките във функцията dequeue .
c) Да се открият, обяснят и коригират грешките във функцията main .

30. Нека е дадено двоичното дърво:




```
struct btree_el   //тип на елементите във върховете на дървото
{ btree_el *left; //указател към ляво поддърво
  char key;
  btree_el *right; //указател към дясно поддърво
};
```

a) Какво ще изведе функцията?

```
void inorder(btree_el *t)
{ if (t != NULL)
     { inorder(t->left);
       cout << t->key << ' ';
       inorder(t->right);
     }
}
```

b) Какво ще изведе функцията?

```
void preorder(btree_el *t)
{ if (t)
     { putchar(t->key);
       preorder(t->left);
       preorder(t->right);
     }
}
```

c) Какво ще изведе функцията?

```
void postorder(btree_el *t)
{ if (t)
     { postorder(t->left);
       postorder(t->right);
       printf("%c%c", t->key, ' ');
     }
}
```

31. Коя от следните формули <u>не е</u> логическо следствие от формулата p & (p ⇒ q) ?

a) p

b) q

c) p & ¬ p

d) p ⇔ q

32. Нека p и q са едноместни предикати, а X и Y са променливи.
Кои от следните формули <u>не съдържат</u> грешки?

a) ∀X(p(X) & ¬q(X))

b) p(q(X)) ∨ r(Y)

c) p(X,Y) & ¬q(X)

d) ∃X(p(¬X) ⇒ q(X))

33. Кои от следните двойки формули са логически еквивалентни?

a) ∀X(p(X) ⇒ q(X)) **и** (∀X p(X)) ⇒ (∀X q(X))

b) ∀X(p(X) ⇒ q(X)) **и** ∀X(¬q(X) ⇒ ¬p(X))

c) ∀X((∃Yp(Y)) ⇒ q(X)) **и** ∀X∀Y(p(Y) ⇒ q(X))

d) ¬(∀X (p(X) ⇒ q(X))) **и** ∃X(q(X) ⇒ p(X))

34. Кои от следните двойки списъци са униформизируеми? (Променливите са означени с главни, а атомите - с малки букви.)
Напишете и съответните стойности на променливите, за които тези двойки списъци стават еднакви.

a)	[X | Y]	**и**	[a]	X =	Y =	
b)	[a | X]	**и**	[a, X]	X =		
c)	[a | X]	**и**	[a, b]	X =		
d)	[[X|Y], Z]	**и**	[[a,b], c ,d]	X =	Y =	Z =

35. Кой от следните предикати е генератор за множеството N = {0,1,2,…} ?
(т.е. при цел ? – nat(X). при многократно преудовлетворяване поражда последователно елементите на N.)

a)
```
nat(0).
nat(X) :- nat(X-1).
```

b)
```
nat(0).
nat(X) :- X is Y+1, nat(Y).
```

c)
```
nat(0).
nat(X) :- nat(Y), X is Y+1.
```

d)
```
nat(0).
nat(X) :- Y is X – 1, nat(Y).
```

36. Кои от следните предикати при цел ? – p(5, 10, Z). (след съответния брой преудовлетворявания)
ще генерират в променливата Z последователно целите числа от интервала [5,10] ?

a)
```
p(X,Y,Z) :- X =< Z, Z =< Y.
```

b)
```
p(X,Y,X) :- X =< Y.
p(X,Y,Z) :- X < Y, X1 is X + 1, p(X1, Y, Z).
```

c)
```
p(X,Y,Z) :- X =< Y.
p(X,Y,Z) :- X < Y, X1 is X + 1, p(X1, Y, Z).
```

d)
```
p(X,Y,Z) :- X > Y, ! , fail.
p(X,Y,X).
p(X,Y,Z) :- X1 is X + 1, p(X1, Y, Z).
```

37. Кои от следните предикати del изтриват <u>всички</u> участия на елемент в списък,
т.е. еднократното извикване на целта ? – del (a, list, Z) . връща в Z списъка list , от който са изтрити всички участия на a .
(Променливите са означени с главни, а атомите - с малки букви.)
(Пример: при ? – del (a, [a,b,c,a], Z). получаваме Z=[b,c])

a)
```
del ( X,  [X | Y], Y ).
del( X, [A | Y], [A | Z]) :- del(X, Y, Z ).
```

b)
```
del ( X,  [ ],  [ ] ) .
del  (X, [X | Y ], Z) :- del(X, Y, Z ).
del( X, [A | Y], [A | Z]) :-  A \ = X,  del(X, Y, Z ).
```

c)
```
del ( X,  [ ],  [ ] ) .
del  (X, [X | Y ], Z) :-  !,   del(X, Y, Z ).
del( X, [A | Y], [A | Z]) :-  del(X, Y, Z
```

d)
```
del ( X,  [X| Y ],  Y ) .
del( X, [A | Y], [A | Z]) :-  A \ = X,  del(X, Y, Z ).
```

38. Да разгледаме предиката mem със следната дефиниция:
(Променливите са означени с главни, а атомите - с малки букви.)

```
mem( X, [ X | _ ].
mem( X, [ _  | Y] :- mem ( X, Y).
mem( X, [ Y | _ ] :- mem ( X, Y).
```

На кои от следните цели PROLOG ще отговори с “yes”?

a) **? – mem ([a], [[a, b]]).**

b) **? – mem (a, [[a]]).**

c) **? – mem ([ ], [a, b]).**

d) **? – mem ([ ], [a, [[ ]]]).**

39. Каква последователност от стойности за променливата X ще генерира предикатът mem от предната задача , при цел ? – mem (X, [a, [b, c]]). в резултат от нейното четирикратно преудовлетворяване ?

a) **X = a, X = [b, c], X = b, X = c**

b) **X = a X = b, X = [b, c], X = c**

c) **X = a, X = b, X = c, X = [b, c]**

d) **X = [b, c], X = a, X = b, X = c**

40. Нека е дадена програмата:

```
student ( john).
married  (bill).
```

На кои от следните цели PROLOG ще отговори с “no”?

a) **?- not (married (john)).**

b) **?- not (married (X)).**

c) **?- not (married (X)), student (X).**

d) **? - student (X), not (married (X)).**

41) Какъв ще бъде резултатът от изпълнението на следната програма на езика Scheme:

```
(define (f x y)
  (if (= x 1) (- x) y))
(define (g) ((lambda () 1)))
(define (h) (h))
(f (g) (h))
```

ако реализацията на интерпретатора се основава на апликативния модел на оценяване?

- **а) съобщение за грешка**
- **б) изпълнението няма да завърши**
- **в) 1**
- **г) –1**

42) Какъв тип изчислителен процес генерира следната програма на езика Scheme:

```
(define (f x n)
  (define (square x) (* x x))
  (cond ((= n 0) 1)
      ((even? n) (square (f x (/ n 2))))
      (else (* x (f x (- n 1))))))
```

- **а) линеен рекурсивен процес**
- **б) линеен итеративен процес**
- **в) логаритмичен рекурсивен процес**
- **г) логаритмичен итеративен процес**

43) Попълнете липсващите изрази в дефиницията на функцията f при условие, че тя пресмята произведението на a и b, като за целта генерира логаритмичен итеративен процес. В липсващите изрази могат да бъдат използвани операциите: събиране, изваждане, умножение с 2, деление на 2.

```
(define (f a b)
  (define (f1 a b p)
    (cond ((= b 0) p)
        ((even? b) (f1 _____ _____ p))
        (else (f1 a _____ _____ ))))
  (f1 a b 0))
```

44) Нека са дадени (т.е. оценени от интерпретатора на Scheme) следните изрази:

```
(define make-mystery
  (lambda (the-count)
    (lambda (x)
      (cond ((null? x) the-count)
            ((number? x) (set! the-count (+ the-count x)) the-count)
            (else (set! the-count (- the-count 1)) the-count)))))
(define mystery (make-mystery 0))
```

Напишете оценките на следващите изрази при условие, че тези изрази се оценяват точно в реда (1), (2), (3):

(1)	**(mystery '())**	**➔**
(2)	**(2) (mystery (+ 1 1))**	**➔**
(3)	**(mystery 'a)**	**➔**

45) Напишете оценката на всеки от следващите изрази на езика Scheme:

(car (cdr '(a (b c) d)))	**➔**
(cdr (car '(((a b) (c d)) (e f))))	**➔**
(cdr (cdr '((a b) ((c d)))))	**➔**

46) Напишете оценката на всеки от следващите изрази на езика Scheme:

(cons 'a (cons 'b (cons 'c nil)))	**➔**
(append (car '((a b) (c d))) (cdr '(x y z)))	**➔**
(list '(a b) (list 'c))	**➔**

47) Напишете оценката на всеки от следващите изрази на езика Scheme:

```
(map list '(a b c))                          ||  ➔
((lambda (x y) (if (< x y) y x)) 3 5)        ||  ➔
(map (lambda (x) (* x x)) (list 1 2 3))      ||  ➔
```

48) Даден е (т.е. оценен е интерпретатора на Scheme) изразът:

```
(define f (lambda (x) (lambda (y) (+ x y))))
```

Напишете какво е действието на функцията f. В частност, каква е оценката на израза (f 10)?

49) Попълнете липсващите изрази в дефиницията на функцията repeated при условие, че тя предизвиква n-кратно прилагане на дадена едноаргументна функция f по следния начин:

```
((repeated f n) x)  -->  f(f( . . .  (f(x))  . . . )).
                          \__ n __/
(define (repeated f n)
  (lambda (x)
    (if ( = n 1)
        (f ____ )
        (f _____________________________ ))))
```

50) Какво е предназначението на следната функция на езика Scheme:

```
(define (f l1 l2)
  (cond ((null? l2) '())
        ((member (car l2) l1) (cons (car l2) (f l1 (cdr l2))))
        (else (f l1 (cdr l2)))))
```

а) намира сечението l1∩l2
б) намира обединението l1∪l2
в) намира разликата l1 \ l2
г) намира разликата l2 \ l1

51) Попълнете липсващия израз в дефиницията на функцията scons при условие, че тя получава като аргументи S-израз a и списък от списъци ll и връща като резултат списък, получен чрез добавяне на a в началото на всеки от елементите на списъка ll:

```
(define (scons a ll)
  (map _____________________________ ll))
```

52) Нека са дадени (т.е. оценени от интерпретатора на Scheme) следните изрази:

```
(define make-mystery
  (lambda (the-list)
    (lambda (x)
      (cond ((null? x) the-list)
            ((atom? x) (set! the-list (cons x the-list)) the-list)
            (else (set! the-list (append x the-list)) the-list)))))
(define mystery (make-mystery '(a)))
```

Напишете оценките на следващите изрази при условие, че тези изрази се оценяват точно в реда (1), (2), (3):

```
(1)   (mystery '())      ||  ➔
(2)   (mystery 'b)       ||  ➔
(3)   (mystery '(c d))   ||  ➔
```

53) Попълнете липсващите изрази в дефиницията на функцията merge при условие, че тя слива два списъка l1 и l2 от числа, които са сортирани във възходящ ред (резултатът също е сортиран във възходящ ред и съвпадащите елементи на l1 и l2 участват в него в толкова екземпляра, колкото пъти се срещат сумарно в l1 и l2):

```
(define (merge l1 l2)
  (cond ((null? l1) l2)
        ((null? l2) l1)
        ((<= (car l1) (car l2))  _____________________________________ )

        (else  _____________________________________ )))
```

Даден е ориентиран граф, чиито възли са именувани с уникални идентификатори. Графът е описан чрез средства на езика Scheme като асоциативен списък, в който ключове са имената на възлите, а асоциираната с даден ключ стойност е списък от имената на преките наследници на съответния възел. Дадени са следните дефиниции на функции:

```
(define (search node1 node2)
  (reverse (srch (list (list node1)) node2)))
(define (srch paths goal)
  (cond ((null? paths) #f)
        ((eq? (car (car paths)) goal) (car paths))
        (else (srch < ??? > goal))))
(define (extend path)
  (apply append (map (lambda (x) (if (memq x path) '() (list (cons x path))))
                     (succs (car path)))))
(define (succs node) (cdr (assq node graph)))
```

54) Попълнете липсващия израз в дефиницията на функцията srch при условие, че тя намира път в графа от възела node1 до възела node2, като за целта реализира стратегия на обхождане в дълбочина.

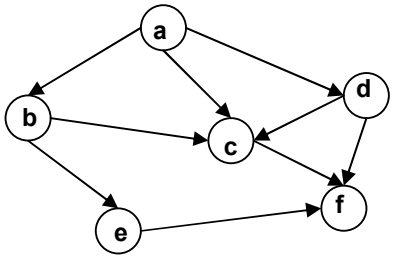
```
(define (srch paths goal)
  (cond ((null? paths) #f)
        ((eq? (car (car paths)) goal) (car paths))
        (else (srch _____ goal))))
```

55) Попълнете липсващия израз в дефиницията на функцията srch при условие, че тя намира път в графа от възела node1 до възела node2, като за целта реализира стратегия на обхождане в широчина.

```
(define (srch paths goal)
  (cond ((null? paths) #f)
        ((eq? (car (car paths)) goal) (car paths))
        (else (srch _____ goal))))
```

56) Даден е ориентиран граф, представен чрез поредица от факти на Prolog по следния начин:

```
arch(a,b).
arch(a,c).
arch(a,d).
arch(b,e).
arch(b,c).
arch(e,f).
arch(c,f).
arch(d,c).
arch(d,f).
```



Какъв път от възела a до възела f ще се получи като резултат, ако се използва стратегията на обхождане на графа в широчина?

а) abcf
б) abef
в) adf
г) acf

57) Даден е ориентиран граф, представен чрез поредица от факти на Prolog от вида arch(<node1>,<node2>) всеки от които означава, че в графа съществува дъга с начало <node1> и край <node2>.
Допишете липсващите цели в дефиницията на отношението extend_b при условие, че с помощта на отношението depth_bound_search се решава задачата за търсене на път Path от даден начален възел Start да целевия възел Goal, като се реализира алгоритъмът за търсене в дълбочина до определено ниво Depth:

```
depth_bound_search(DepthBound,Start,Goal,Path):-
  depth_bound(DepthBound,[[Start]],Goal,L), reverse(L,Path).
depth_bound(_,[[Goal|Path]|_],Goal,[Goal|Path]).
depth_bound(Depth,[Path|Queue],Goal,FinalPath):-
  extend_b(Depth,Path,NewPaths),
  append(NewPaths,Queue,NewQueue),
  depth_bound(Depth,NewQueue,Goal,FinalPath).
extend_b(Depth,[Node|Path],NewPaths):- length(Path,Len), _____ , !,
                                                                  _____ .
extend_b(_,_,[]).
```

58) Даден е ориентиран граф, представен чрез поредица от факти на Prolog от вида arch(<node1>,<node2>,<dist>), всеки от които означава, че в графа съществува дъга с начало <node1>, край <node2> и дължина <dist>.
Какъв алгоритъм за търсене на път Path от даден начален възел Start да целевия възел Goal се реализира чрез дефинираното по-долу отношение search:

```
search(Start,Goal,Path):- srch([[Start]],Goal,L), reverse(L,Path).
srch([[Goal|Path]|_],Goal,[Goal|Path]).
srch([Path|Queue],Goal,FinalPath):- extend(Path,NewPaths),
  append(Queue, NewPaths,Queue1), sort_queue(Queue1,NewQueue),
  srch(NewQueue,Goal,FinalPath).
sort_queue(L,L2):- change(L,L1), !, sort_queue(L1,L2).
sort_queue(L,L).
change([X,Y|T],[Y,X|T]):- path_cost(X,CX), path_cost(Y,CY), CX>CY.
change([X|T],[X|T1]):- change(T,T1).
path_cost([A,B],Cost):- arch(B,A,Cost).
path_cost([A,B|T],Cost):- arch(B,A,Cost1), path_cost([B|T],Cost2), Cost is Cost1+Cost2.
```

а) търсене на най-добър път (best-first search)
б) търсене с равномерна цена на пътя (uniform cost search)
в) търсене по метода на най-бързото изкачване (hill climbing)
г) търсене по метода A*

Базата от правила и работната памет (базата от факти) на една система, основана на правила, са описани със средствата на езика Prolog както следва:

```
% Дефиниции на използваните оператори
:- op(900,fx,if).
:- op(890,xfx,then).
:- op(880,xfy,or).
:- op(870,xfy,and).
% Правила
if has_hair(X) or gives_milk(X) then mammal(X).
if can_fly(X) then bird(X).
if can_swim(X) then fish(X).
if mammal(X) and can_speak(X) then human(X).
if bird(X) and can_speak(X) then parrot(X).
if fish(X) and has_sharp_teeth(X) then shark(X).
% Факти (работна памет)
fact(has_hair(gogo)).
fact(can_fly(pepo)).
fact(can_swim(sharky)).
fact(can_speak(gogo)).
fact(can_speak(pepo)).
fact(has_sharp_teeth(sharky)).
```

Предполага се, че интерпретаторът на правилата извършва прав извод (извод, управляван от данните), като на всяка стъпка от работния си цикъл записва заключението на избраното за изпълнение правило като нов факт в края на работната памет. В конфликтното множество се включват само правила, при чието изпълнение се записват нови факти в работната памет.

59) Какво ще съдържа работната памет след завършване на работата на интерпретатора на правилата, ако на всяка стъпка от работния си цикъл той избира и изпълнява първото правило от конфликтното множество?

60) Какво ще съдържа работната памет след завършване на работата на интерпретатора на правилата, ако на всяка стъпка от работния си цикъл той избира и изпълнява това правило от конфликтното множество, чието условие се удовлетворява от най-скоро записан в работната памет факт?

61) Допишете дефиницията на отношението ok, ако отношението derive играе ролята на интерпретатор на правилата, който извършва прав извод, като на всяка стъпка от работния си цикъл избира и изпълнява първото правило от конфликтното множество:

```
derive:- derive_fact(P), write('Derived: '), write(P), nl, derive.
derive.
derive_fact(P):- if Condition then P, ok(Condition), \+ fact(P), assertz(fact(P)).
ok(P):- fact(P).
ok( _____ ):- _____ , _____ .
ok( _____ ):- _____ , _____ .
```

62) Дадена е семантична мрежа, която е представена чрез поредица от факти на Prolog от вида <relation>(<argument1>,<argument2>). С имената на релациите са означени дъгите на мрежата, а с имената на обектите, които са аргументи на релациите, са означени възлите на семантичната мрежа. Релацията isa служи за означаване на връзки от тип обект – клас и клас – суперклас между аргументите. С помощта на отношението deduce(<relation>(<argument1>,<argument2>)) се реализира интерпретатор на семантичната мрежа, който е предназначен за извличане на знания за обектите, описани чрез мрежата. Попълнете липсващите цели в дефиницията на deduce:

```
deduce(Fact):- call(Fact), !.
deduce(Fact):- Fact =.. [Rel,Arg1,Arg2], isa(Arg1,SuperArg),
               _____ , _____ .
```

Фреймовете в дадена фреймова система имат следната структура:

```
(<frame-name> (<slot-1> (<facet-11> <value-11>)
                        (<facet-12> <value-12>)
                                . . . )
              (<slot-2> (<facet-21> <value-21>)
                        (<facet-22> <value-22>)
                                . . . )  . . .  )
```

В базата от знания на системата са дефинирани следните фреймове:

```
(human (height (default 170)) (hobbies (default (football cinema)))
       (favorite_books (value “Gone with the Wind”)))
(student (hobbies (default (reading TV))) (favorite_books (default “I, Robot”))
       (isa (value human)))
(bobo (isa (value student)) (height (value 180)))
```

63) Какъв ще бъде резултатът от Z-търсенето в посочената база от фреймове за слота favorite_books на фрейма bobo?

64) Какъв ще бъде резултатът от N-търсенето в посочената база от фреймове за слота favorite_books на фрейма bobo?