p3.

or) program

Declarations {
```
int a, b, c;
double d;
```
) → build symbol table

```
begin
    b = 3*(5+2);
    c = (3+4)*5;
    print b;
    print c;
    print (2+3)*7+2^3;
end
```
Statements { (covers the block above)

— main()
```
{ read a word;
  if (word == "program")
      [ Call Declarations();
        Call Statements(); ]
  else
      error; exit(1);
}
```

build symtab

Declarations() — recursive
```
{ read a word;
  if (word == "begin")
      exit(1);
  else ("int" or "double")
      Call Declaration();   ← word (int/dou...)
  Call Declarations();
}
```
ok

int/double

— Declaration (word)
```
{ read id;
  Store it in symbol_table[index].id;
  Store word in symbol_table[index].type;
  index++;
  loop until (;) meets;
}
```
global or local

read a char:
i.e., [if (;), repeat
        else (;) — exit(1)]

Symbol_table[ ] — <char, stry, in...

id	type	value
0	a	int	
1	b	int	21
2	c	int	35
⋮	d	double	

struct node
```
{ char id;
  strp type;
  int val;
}
struct node
symbol_table[...]
```

symbol_table[index].id = a;
symbol_table[index].type = "int";
symbol_table[index].val = 21;

```
— Statements ()  — recursive
  { read a word;
      if (word == "end")
          exit(1);
      else
          call Statement (word);
    call Statements ();
  }
```

```
— Statement (word)
  { if (word == "print")
        call print_St();
    else (id)
        call Assign_St (word);
  }
```
(id)

extract val. from sym-tab.

```
— print_St ()
  { read a char (id);
    if (id is 'a'~'z')
        search symbol-tab for id
        and get index;
        if (found)
            display symbol-tab[index].val;
        else
            semantic error;
    else (expression case)
        get back one position;
        call Exp() and display result;
  }
```

enter val. to sym-tab

```
— Assign_St (word)          id
  { read a word;
    if (word == "=")   char
        int temp = Exp();
        search symbol-tab for word
        and get index;
        if (found)
            symbol-tab[index].val = temp
        else
            semantic error;
    else
        syntax error.
  }
```