TDT 4205
Problem Set 5

The deadline for this problem set is friday, November 9th. Submissions will only be accepted through *It's learning*.

1 Optimizations (10%)

Consider the basic block

```
a=1
b=2
c=3
d=a+x
e=b+c
f=e
g=f
g=d+y
a=b+c
```

Each of the following basic blocks is the result of applying an optimization to the one above. State the name of the optimization in each case.

1.1

```
a=1
b=2
c=3
d=a+x
e=b+c
f=e
g=d+y
a=b+c
```

1.2

```
a=1
b=2
c=3
d=a+x
e=b+c
t1=e
f=e
g=f
g=d+y
a=t1
```

1.3

```
a=1
b=2
c=3
d=1+x
e=5
f=5
g=5
g=d+y
a=5
```

1.4

```
a=1
b=2
c=3
d=a+x
e=b+c
f=e
g=e
g=d+y
a=b+c
```

2 Practical work

These tasks expand upon last week's work by implementing a symbol table in *symtab.c*.

You may note that the tree node structure in *tree.h* has been expanded to include a pointer to the newly introduced `symtab_t` structure for symbol table entries: the purpose of this exercise is to use the string manipulation functions from PS4 to adapt the `find_symbols` function, such that it introduces `symtab_t` structures in a hash table at variable declarations, and to bind occurrences of these variables in the tree to the correct entry through the `entry` pointer.

The purpose of the element `char *label` is merely to provide storage such that the correctness of these bindings can be written during debugging. The element `int stack_offset` is as yet unused, and does not have to be set to any particular value in this exercise.

2.1 30%

Implement the functions

```
void create_symtab ( void )
void destroy_symtab ( void )
void destroy_symtab_entry ( symtab_t *discard )
void enter_symbol ( char *key, symtab_t *value )
symtab_t * lookup_symbol ( char * key )
```

in *symtab.c*, using a hash table created by `hcreate` from `<search.h>`.

`create_symtab` should allocate a hash table with space for a number of entries defined by the macro `TABLE_SIZE`.

`destroy_symtab` should free the memory associated with both the table and its entries.

`destroy_symtab_entry` should free all memory associated with the given table entry.

`enter_symbol` should place an entry in the table, indexed by the provided string.

`lookup_symbol` should return a pointer to the entry indexed by the provided string if one exists, or NULL otherwise.

2.2 20%

Expand `find_symbols` to enter `symtab_t` structures in the symbol table for declarations of functions and variables.

2.3 20%

Write a function which, given a tree node representing a variable or function call as well as a scope string like those generated in PS4, recursively unwinds the scopes on the stack, testing if there is an entry for the given variable in each scope. If one is found recursion should terminate, so that this function effectively finds the closest nested declaration for the occurrence of a variable. Otherwise, NULL should be returned.

2.4 20%

Expand `find_symbols` to bind occurrences of variables and function names to their corresponding symbol table entries through the `entry` pointer.