Reasoning About Code

1/25/2010

```
int deref(int *p) {
    return *p;
}
```

```
/* requires: p != NULL */
int deref(int *p) {
    return *p;
}
```

```
int sum(int a[], size_t n) {
  int total = 0;
  for (size_t i=0; i<n; i++)
    total += a[i];
  return total;
}
```

```
/* requires: a != NULL && size(a) >= n */
int sum(int a[], size_t n) {
  int total = 0;
  for (size_t i=0; i<n; i++)
    total += a[i];
  return total;
}
```

```
/* requires: a != NULL && size(a) >= n */
int sum(int a[], size_t n) {
  int total = 0;
  for (size_t i=0; i<n; i++)

    total += a[i];
  return total;
}
```

```
/* requires: a != NULL && size(a) >= n */
int sum(int a[], size_t n) {
  int total = 0;
  for (size_t i=0; i<n; i++)
      /* 0 <= i && i < n && n <= size(a) */
    total += a[i];
  return total;
}
```

```
int sumderef(int *a[], size_t n) {
    int total = 0, i;
    for (i=0; i<n; i++)
         total += *(a[i]);
    return total;
}
```

Woops! If (int)n < 0, i becomes negative, and a[i] is unsafe.

```
int sumderef(int *a[], size_t n) {
    int total = 0, i;
    for (i=0; i<n; i++)
         total += *(a[i]);
    return total;
}
```

```c
int sumderef(int *a[], size_t n) {
    int total = 0;
    for (size_t i=0; i<n; i++)
         total += *(a[i]);
    return total;
}
```

```
/* requires: a != NULL &&
     size(a) >= n &&
            ???                         */
int sumderef(int *a[], size_t n) {
    int total = 0;
    for (size_t i=0; i<n; i++)
         total += *(a[i]);
    return total;
}
```

```
/* requires: a != NULL &&
     size(a) >= n &&
     for all j in 0..n-1, a[j] != NULL */
int sumderef(int *a[], size_t n) {
    int total = 0;
    for (size_t i=0; i<n; i++)
         total += *(a[i]);
    return total;
}
```

```
void *mymalloc(size_t n) {
    void *p = malloc(n);
    if (!p) { perror("malloc"); exit(1); }
    return p;
}
```

```
/* ensures: retval != NULL */
void *mymalloc(size_t n) {
    void *p = malloc(n);
    if (!p) { perror("malloc"); exit(1); }
    return p;
}
```

```
char *tbl[N];


int hash(char *s) {
  int h = 17;
  while (*s)
    h = 257*h + (*s++) + 3;
  return h % N;
}

bool search(char *s) {
  int i = hash(s);
  return tbl[i] && (strcmp(tbl[i], s)==0);
}
```

```
char *tbl[N];

/* ensures: 0 <= retval && retval < N */
int hash(char *s) {
  int h = 17;
  while (*s)
    h = 257*h + (*s++) + 3;
  return h % N;
}

bool search(char *s) {
  int i = hash(s);
  return tbl[i] && (strcmp(tbl[i], s)==0);
}
```

```
char *tbl[N];

/* ensures: 0 <= retval && retval < N */
int hash(char *s) {
  int h = 17;                 /* 0 <= h */
  while (*s)
    h = 257*h + (*s++) + 3;
  return h % N;
}

bool search(char *s) {
  int i = hash(s);
  return tbl[i] && (strcmp(tbl[i], s)==0);
}
```

```
char *tbl[N];

/* ensures: 0 <= retval && retval < N */
int hash(char *s) {
  int h = 17;                 /* 0 <= h */
  while (*s)                  /* 0 <= h */
    h = 257*h + (*s++) + 3;
  return h % N;
}

bool search(char *s) {
  int i = hash(s);
  return tbl[i] && (strcmp(tbl[i], s)==0);
}
```

```c
char *tbl[N];

/* ensures: 0 <= retval && retval < N */
int hash(char *s) {
  int h = 17;                 /* 0 <= h */
  while (*s)                  /* 0 <= h */
    h = 257*h + (*s++) + 3;   /* 0 <= h */
  return h % N;
}

bool search(char *s) {
  int i = hash(s);
  return tbl[i] && (strcmp(tbl[i], s)==0);
}
```

```
char *tbl[N];

/* ensures: 0 <= retval && retval < N */
int hash(char *s) {
  int h = 17;                 /* 0 <= h */
  while (*s)                  /* 0 <= h */
    h = 257*h + (*s++) + 3;   /* 0 <= h */
  return h % N; /* 0 <= retval < N */
}

bool search(char *s) {
  int i = hash(s);
  return tbl[i] && (strcmp(tbl[i], s)==0);
}
```

```
char *tbl[N];

/* ensures: 0 <= retval && retval < N */
int hash(char *s) {
  int h = 17;                 /* 0 <= h */
  while (*s)                  /* 0 <= h */
    h = 257*h + (*s++) + 3;   /* ~~0 <= h~~ */
  return h % N; /* 0 <= retval < N */
}

bool search(char *s) {
  int i = hash(s);
  return tbl[i] && (strcmp(tbl[i], s)==0);
}
```

```
char *tbl[N];

/* ensures: 0 <= retval && retval < N */
int hash(char *s) {
  int h = 17;                  /* 0 <= h */
  while (*s)                   /* 0 <= h */
    h = 257*h + (*s++) + 3;    /* ~~0 <= h~~ */
  return h % N; /* ~~0 <=~~ retval < N */
}

bool search(char *s) {
  int i = hash(s);
  return tbl[i] && (strcmp(tbl[i], s)==0);
}
```

```
char *tbl[N];

/* ensures: ~~0 <= r~~etval && retval < N */
int hash(char *s) {
  int h = 17;                 /* 0 <= h */
  while (*s)                  /* 0 <= h */
    h = 257*h + (*s++) + 3;   /* ~~0 <= h~~ */
  return h % N; /* ~~0 <= r~~etval < N */
}

bool search(char *s) {
  int i = hash(s);
  return tbl[i] && (strcmp(tbl[i], s)==0);
}
```

```
char *getcomment(char *src, size_t srclen) {
    size_t n = (src[0]<<8) + src[1];
    size_t clen = n – 2;
    char *comment = malloc(clen+1);
    memcpy(comment, src, clen);
    comment[clen] = ‘\0’;
    return comment;
}
```