Problem 1

- Complex numbers are of the form $a + i\ b$.

$\Rightarrow$ Complex numbers can be regarded as tuples of the form (a, b).

- Programming feature / construct that is used to represent tuples (or any group of values that together form a meaningful unit): *struct*

```
typedef struct {
  float real, img;
} COMPLEX;

COMPLEX root1, root2;
```

Problem 3 – I

```c
#include "common.h"

#define BUF_LEN 256
#define FORMAT_STRING "%-16s %-40s %-10s\n"

int main(int ac, char *av[])
{
    char inbuf[BUF_LEN], *roll, *name, *uid, *cptr;
    FILE *fp;

    if (NULL == (fp = fopen("etc-passwd.txt", "r")))
        ERR_MESG("main: error opening file");

    /* Each line in /etc/passwd is of the form:
     * <roll number>:x:<uid>:<gid>:<name>:<home directory>:<login shell>
     */
    printf(FORMAT_STRING, "ROLL", "NAME", "UID");
    while (NULL != fgets(inbuf, BUF_LEN, fp)) {
        roll = inbuf;
        for (cptr = inbuf; *cptr != ':'; cptr++);
        *cptr = '\0';
```

Problem 3 – II

```
        cptr += 3; // skip over x and point to UID
        for (uid = cptr; *cptr != ':'; cptr++);
        *cptr = '\0';

        while (*cptr++ != ':'); // skip over gid
        for (name = cptr; *cptr != ':'; cptr++);
        *cptr = '\0';

        printf(FORMAT_STRING, roll, name, uid);
    }

    fclose(fp);
    return 0;
}
```

Problem 5

```
unsigned int strlen(char *s) {
  int i;
  for (i = 0; s[i]; i++);
  return i;
}

unsigned int strcmp(char *s, char *t) {
  do {
     if (*s != *t) return 0;
  } while (*s++ && *t++);
  return 1;
}

unsigned int diffByOne(char *s, char *t) {
  int numDiffs = 0;
  while (*s && *t) {
     if (*s++ != *t++) numDiffs++;
  }
  if (*s != *t || numDiffs != 1) return 0;
  return 1;
}
```

Problem 6

```
int uniquify(unsigned int *a, unsigned int n) {
    int i, j;
    if (n==0) return 0;
    for (i = 0, j = 1; j < n; j++) {
        if (a[j] < a[i])
            return -1; // array is not sorted in increasing order
        if (a[j] > a[i]) // a[j] is the next distinct integer
            a[++i] = a[j];
    }
    return i+1;
}
```

Problem 7

Suppose that we choose to represent sets by arrays of non-negative integers, containing at most 100 elements each. Let A and B be 2 such sets of integers. Implement the following set operations:

1. $A \cup B$;
2. $A \cap B$;
3. $A \subseteq B$ (returns 1 if A is a subset of B, 0 otherwise);
4. prints the power set of A on the screen.

Solution coming soon.

Problem 8 – I

```c
#include "common.h"
#include <sys/types.h>
#include <sys/stat.h>
#include <unistd.h>

static int
compare_strings(void *p1, void *p2)
{
    char *s1, *s2;
    s1 = *((char **) p1);
    s2 = *((char **) p2);
    return strcmp(s1,s2);
}

int main(int ac, char *av[])
{
    char *inbuf, **lines;
    int filesize, num_lines, i;
```

Problem 8 – II

```
FILE *fp;
struct stat statbuf;

if (ac != 2)
    ERR_MESG("Usage: sort <filename>");

/* Get file size */
if (NULL == (fp = fopen(av[1], "r")) ||
    -1 == fstat(fileno(fp), &statbuf)) // see man 2 stat
    ERR_MESG("sort: error opening / stat-ing file");
filesize = statbuf.st_size;

/* Read in the whole file.
 * Keep an extra '\0' as sentinel at end of file to correctly
 * handle files that do not end with '\n'
 */
if (NULL == (inbuf = Malloc(filesize + 1, char)))
    ERR_MESG("sort: out of memory");
inbuf[filesize] = '\0';
```

Problem 8 – III

```
if (filesize != fread(inbuf, sizeof(char), filesize, fp))
    ERR_MESG("sort: error reading file");
fclose(fp);

/* Count number of lines */
for (i = 0, num_lines = 0; i < filesize; i++)
    if (inbuf[i] == '\n') num_lines++;
if (inbuf[i-1] != '\n') // file did not end with newline
    num_lines++;
else inbuf[i-1] = '\0';
/* Set up pointers from lines to inbuf as shown in diagram */
if (NULL == (lines = Malloc(num_lines, char *)))
    ERR_MESG("sort: out of memory");
for (i = 0, num_lines = 0, lines[num_lines++] = inbuf; i <
     filesize; i++) {
    if (inbuf[i] == '\n') {
        inbuf[i] = '\0';
        lines[num_lines++] = inbuf + i + 1;
    }
```

Problem 8 – IV

```
    }

#if 0
    /* Sanity check: has the file been read correctly? */
    for (i = 0; i < num_lines; i++) {
        puts(lines[i]);
    }
#endif

    /* Sort and print */
    qsort((void *) lines, num_lines, sizeof(char *), compare_strings);
    for (i = 0; i < num_lines; i++) {
        puts(lines[i]);
    }

    return 0;
}
```