```c
/**************************************
 * dma.c
 * Doug Lloyd
 * October 3, 2010
 *
 * Fun with dynamic memory allocation
 **************************************/

/* Header files */
#include <stdio.h>
#include <cs50.h>
#include <stdlib.h> // for malloc() and free()

/* Function Definitions */
int main() {

  // Get an integer
  printf("Please input an integer: ");
  int i = GetInt();

  // Make an array that size, statically
  int arr1[i];

  // Assign its contents, sequentially
  for(int a = 0; a < i; a++)
    arr1[a] = a;

  // Make another array that size, dynamically
  int *arr2 = (int *) malloc(sizeof(int) * i);

  // Assign its contents, sequentially
  for(int a = 0; a < i; a++)
    *(arr2 + a) = a;

  // Print them both side-by-side
  // Notice interoperability of pointers and array syntax
  for(int a = 0; a < i; a++) {
    printf("arr1[%d] = %d; *(arr2 = %d) = %d\n",
           a, *(arr1 + a), a, arr2[a]);
  }

  // Give memory back!
  free(arr2);
  return 0;
}

```

```c
/***************************************
 * pointers.c
 * Doug Lloyd
 * October 3, 2010
 *
 * Fun with pointers
 ***************************************/

/* Header files */
#include <stdio.h>
#include <cs50.h>

/* Function Definitions */
int main() {

  // Get an integer
  printf("Please input an integer: ");
  int i = GetInt();

  // Show its location
  printf("That integer is located at memory address: %X\n", &i);

  // Create a pointer, have it point to i
  int *pi;
  pi = &i;

  // Show its contents
  printf("The value of pi, the pointer to i, is: %X\n", pi);
  printf("Which means that *pi is: %d\n", *pi);

  // Get another integer
  printf("Change the value of i by how much: ");
  int j = GetInt();

  // Change it by way of the pointer
  *pi += j;

  printf("The new value of i is: %d\n", *pi);

  return 0;
}


```

```
/**************************************
 * struct.c
 * Doug Lloyd
 * October 3, 2010
 *
 * Fun with structs
 ***************************************/

/* Header files */
#include <stdio.h>
#include <cs50.h>
#include <unistd.h>

/* Structure Declarations */
struct cat_t {
  string name;
  int age;
  char gender;
};

/* Function Declarations */
struct cat_t makeCat(string n, int a, char g);
void printCat(struct cat_t c);

/* Function Definitions */
int main() {

  // Get some info
  printf("What is your cat’s name? ");
  string name = GetString();
  char gender;
  do {
    printf("And is it a male (M) or a female (F)? ");
    gender = GetChar();
  } while(gender != ’M’ && gender != ’F’);
  string prompt = (gender == ’M’) ? "he" : "she";
  printf("Lastly, how old is %s? ", prompt);
  int age = GetInt();

  printf("Thanks. I’ll make a record for your cat now\n");
  sleep(1);
  printf("Making record...\n");
  struct cat_t mycat = makeCat(name, age, gender);
  sleep(1);
  printf("Record complete!\n");
  printCat(mycat);
  return 0;
}

struct cat_t makeCat(string n, int a, char g) {
  struct cat_t xcat;
  xcat.name = n;
  xcat.age = a;
  xcat.gender = g;
  return xcat;
}

void printCat(struct cat_t c) {
  printf("\nName: %s", c.name);
  printf("\nAge: %d", c.age);
  printf("\nGender: %c\n", c.gender);
  return;
}

```

```c
/***************************************
 * structdma.c
 * Doug Lloyd
 * October 3, 2010
 *
 * Fun with dynamically-allocated
 * pointers to structs
 ***************************************/

/* Header files */
#include <stdio.h>
#include <cs50.h>
#include <unistd.h>
#include <stdlib.h>

/* Structure Declarations */
struct cat_t {
  string name;
  int age;
  char gender;
};

/* Function Declarations */
void makeCat(struct cat_t *xcat, string n, int a, char g);
void printCat(struct cat_t *c);

/* Function Definitions */
int main() {

  // Get some info
  printf("What is your cat’s name? ");
  string name = GetString();
  char gender;
  do {
    printf("And is it a male (M) or a female (F)? ");
    gender = GetChar();
  } while(gender != ’M’ && gender != ’F’);
  string prompt = (gender == ’M’) ? "he" : "she";
  printf("Lastly, how old is %s? ", prompt);
  int age = GetInt();

  printf("Thanks. I’ll make a record for your cat now\n");
  sleep(1);
  printf("Making record...\n");
  struct cat_t *mycat = malloc(sizeof(struct cat_t));
  makeCat(mycat, name, age, gender);
  sleep(1);
  printf("Record complete!\n");
  printCat(mycat);
  free(mycat);
  return 0;
}

void makeCat(struct cat_t *xcat, string n, int a, char g) {
  xcat->name = n;
  xcat->age = a;
  xcat->gender = g;
  return;
}

void printCat(struct cat_t *c) {
  printf("\nName: %s", c->name);
  printf("\nAge: %d", c->age);
  printf("\nGender: %c\n", c->gender);
  return;
}

```

```c
/***************************************
 * typedef.c
 * Doug Lloyd
 * October 3, 2010
 *
 * Fun with typedef
 ***************************************/

/* Header files */
#include <stdio.h>
#include <cs50.h>
#include <unistd.h>

/* Structure Declarations */
typedef struct _cat_t {
  string name;
  int age;
  char gender;
} cat_t;

/* Function Declarations */
cat_t makeCat(string n, int a, char g);
void printCat(cat_t c);

/* Function Definitions */
int main() {

  // Get some info
  printf("What is your cat’s name? ");
  string name = GetString();
  char gender;
  do {
    printf("And is it a male (M) or a female (F)? ");
    gender = GetChar();
  } while(gender != ’M’ && gender != ’F’);
  string prompt = (gender == ’M’) ? "he" : "she";
  printf("Lastly, how old is %s? ", prompt);
  int age = GetInt();

  printf("Thanks. I’ll make a record for your cat now\n");
  sleep(1);
  printf("Making record...\n");
  cat_t mycat = makeCat(name, age, gender);
  sleep(1);
  printf("Record complete!\n");
  printCat(mycat);
  return 0;
}

cat_t makeCat(string n, int a, char g) {
  cat_t xcat;
  xcat.name = n;
  xcat.age = a;
  xcat.gender = g;
  return xcat;
}

void printCat(cat_t c) {
  printf("\nName: %s", c.name);
  printf("\nAge: %d", c.age);
  printf("\nGender: %c\n", c.gender);
  return;
}

```