# **Lab Sheet 5**

**Exercise 1**

Write a function that makes a copy of an array of integers. Write a test driver for it.

**Exercise 2**

Update the previous test driver to call the copy array function with a size argument that is much larger than the destination array size. Observe the program behavior on execution.

**Exercise 3**

Convert the program in this listing to make use of pointer syntax.

#include <stdio.h>

#define SIZE 10

int sum(int ar[], int n);

int main(void) {

int marbles[SIZE] = {20, 10, 5, 39, 4, 16, 19, 26, 31, 20};

long answer = sum(marbles, SIZE);

printf("The total number of marbles is %ld.\n", answer);

printf("The size of marbles is %zu bytes.\n", sizeof marbles);

return 0;

}

int sum(int ar[], int n) {

int total = 0;

for (int i = 0; i < n; i++)

total += ar[i];

printf("The size of ar is %zu bytes.\n", sizeof ar);

return total;

}

**Exercise 4**

Make this function more compact by combining the pointer increment and dereference operations in the same expression.

int sum(int \*start, int \*end) {

int total = 0;

while (start < end) {

total += \*start; /\* add value to total \*/

start++; /\* advance pointer to next element \*/

}

return total;

}

**Exercise 5**

Write a program that pre-computes the function f(x,y,z)=x+6y+7.2z in a 3D array for all independent variable value combinations ranging from 0 to 100.

**Exercise 6**

Write a program that requests and stores a sequence of input strings, then prompts the user to enter a suffix to be appended to selected strings.

**Exercise 7**

Write a program that accepts IPv4 32-bit addresses in octet form (e.g., 192.168.23.24) and then prints them out both in octet form as well as a whole address value in hex. Make use of unions.

**Exercise 8**

Write a program that fills an employee (in-memory) database with name, surname, ID, monthly salary, and from which to produce two indexes: one for printing the employee list in ascending order according to their ID, and the other by surname. Test the indexes.

**Exercise 9**

Convert this listing to make use of string\_t and name\_t types using typedef as per definitions given in the notes.

#include <stdio.h>

#include <string.h>

#define NLEN 30

struct namect {

char fname[NLEN];

char lname[NLEN];

int letters;

};

struct namect update\_info(struct namect info) {

info.letters = strlen(info.fname) + strlen(info.lname);

return info;

}

int main(void) {

struct namect person = {"Neville", "Grech", 0};

person = update\_info(person);

printf("%s %s, your name contains %d letters.\n",

person.fname, person.lname, person.letters);

return 0;

}

**Exercise 10**

Write a program that accepts two numeric command-line arguments and then adds them up, returning an error message on incorrect usage.