CSCE 212

SHREYAS SRINIVASA
UIN 425 91187

ASSIGNMENT 2

Ex. 2.10

1. Ans. Tested it. ✓

2 Ans: Largest unsigned integer is 4 x 8 = 32 bits.

3. Ans: printf ("%#010x represents the unsigned decimal integer %d \n", bitPattern, bitPattern);

Any integer greater than 7fffffff gives a negative output.

4. Ans:

```
# include <stdio.h>
int main (void)
{
    int x;
    unsigned int y;
    while (1)          // loop "forever"
    {
        printf ("Enter a decimal integer (0 to quit): ");
        scanf ("%i", &x);
        if (x==0) break;    // break out of loop
        printf ("Enter a bit pattern in hexadecimal: ");
        scanf ("%x", &y);
        if (y==0) break;    // break out of loop
        printf ("%i is stored as %#010x or %p,
                and \n", x, x, &x);
        printf ("%#010x represents the decimal integer
                %d stored at %p \n\n", y, y,
                &y);
    }
    printf ("End of program. \n");
    return 0;
}
```

The compiler allocates 2 or 4 bytes depending on CPU architecture.

Exc. 2.14

1.Ans:

```
#include <stdio.h>
int main (void)
{
    char *stringPtr = "Hello world.\n";
    while (*stringPtr != '\0')
    {
        printf ("%p: ", stringPtr);
        printf ("0x%02x\n", *stringPtr);
        stringPtr++;
    }
    printf ("%p: ", stringPtr);
    printf ("0x%02x\n", *stringPtr);
    return 0;
}
```

Ex 2.12

1.Ans:

```
#include <stdio.h>
int main (void)
{
    int anInt = 19088743;
    float aFloat = 19088.743;
    printf ("The integer is %d and the float is %f\n", anInt,
            aFloat);
    return 0;
}
```

The variables are stored at their allocated memory addresses 0x7fffffffde48 and 0x7fffffffde4c. The address of operator (&) along with the print command can be used to print the address of a variable. The %x command can be used to display the values in their stored data type. The command

shows that the integer 19088743 is stored at its respective memory address and the float value 19088.7422 is stored at its respective memory address. Both the values can be displayed in hexadecimal using the correct format letter and when the bytes stored in memory are displayed separately, they appear to follow the little endian storage order. The least significant byte is stored in the lowest numbered address and the smallest byte comes first in memory.

Ex 2.16

1. Ans.

```
#include <unistd.h>
#include <string.h>
int main(void)
{
    char astring[200];
    char *stringPtr = astring;
    write(STDOUT_FILENO, "Enter a text string: ",
          strlen("Enter a text string: "));  // prompt user
    read(STDIN_FILENO, stringPtr, 1); // get first character
    while (*stringPtr != '\n') // look for end of line
    {
        stringPtr++;      // move to next location
        read(STDIN_FILENO, stringPtr, 1); // get next character
    }

    // now echo for user
    write(STDOUT_FILENO, "You entered: \n",
          strlen("You entered: \n"));
    stringPtr = astring;
    do
    {
        write(STDOUT_FILENO, stringPtr, 1);
        stringPtr++;
    } while (*stringPtr != '\n');
    write(STDOUT_FILENO, stringPtr, 1);
    return 0;
}
```

2. Ans:-

```
#include <stdio.h>
#include <unistd.h>
#include <string.h>

int main(void)
{
    char aString[200];
    char *stringPtr = aString;
    write(STDOUT_FILENO, "Enter a test string : ",
          strlen("Enter a test string : "));  // prompt user
    read(STDIN_FILENO, stringPtr, 1);  // get first character
    while (*stringPtr != '\n')  // look for end of line
    {
        stringPtr++;  // move to next location
        read(STDIN_FILENO, stringPtr, 1);  // get next character
    }
    *stringPtr = '\0';  // make into C string
    // now echo for user
    printf("You entered = \n%s\n", aString);
    return 0;
}
```

3. Ans:- There are five separate files for this solution.

```
#include "readstr.h"
#include "writestr.h"
#define STRLEN 5  // limited to 5 for testing readstr
                  // change to 200 for use
int main(void)
{
    char aString[STRLEN];
    writestr("Enter a test string : ");
    readstr(aString, STRLEN);
    writestr("You entered: \n");
    writestr(aString);
    writestr("\n");
    return 0;
}
/* writestr.h
 * Writes a line to standard out.
```


P.T.O

```
 *
 * input:
 *     pointer to C-style text string
 * output:
 *     to screen
 *     returns number of chars written
 */
#ifndef WRITESTR_H
#define WRITESTR_H
int writestr(char *);
#endif
/* writestr.c
 * Writes a line to standard out.
 *
 * input:
 *     pointer to C-style text string
 * output:
 *     to screen
 *     returns number of chars written
 */
#include <unistd.h>
#include "writestr.h"
int writestr(char *stringAddr)
{
    int count = 0;
    while (*stringAddr != '\0')
    {
        write (STDOUT_FILENO, stringAddr, 1);
        count++;
        stringAddr++;
    }
    return count;
}
/* readLn.h
 * Reads a line from standard in.
 * Drops newline character. Eliminates
 * excess characters from input buffer.
 *
 * input from keyboard
```

```
 * output:
 *      null-terminated text string
 *      returns number of chars in text string
 */

#ifndef READLN_H
#define READLN_H
int readln(char *, int);
#endif
/* readln.c
 * Reads a line from standard in.
 * Drops newline character. Eliminates
 * excess characters from input buffer.
 *
 * input:
 *      from keyboard
 * output:
 *      null-terminated text string
 *      returns number of chars in text string
 */
#include <unistd.h>
#include "readln.h"
int readln(char *stringAddr, int maxLength)
{
    int count = 0;
    maxLength--;  // allow space for NUL
    read(STDIN_FILENO, stringAddr, 1);
    while (*stringAddr != '\n')
    {
        if (count < maxLength)
        {
            count++;
            stringAddr++;
        }
        read(STDIN_FILENO, stringAddr, 1);
    }
    *stringAddr = '\0';  // terminate string
    return count;
}
```