DEPARTMENT OF COMPUTER ENGINEERING
CMPE112: Programming Fundamentals
EXPERIMENT 7
Introduction to C Programming: Pointers and string

Objectives:
1) Understand how to edit, compile and execute C computer codes.
2) Understand C programming: Pointers and string

Note: Before writing a computer code, you should do the following steps: 1) **understand** and **analyze** the problem, 2) develop an **algorithm** and/or **flowchart** and 3) convert the **algorithm** and/or the **flowchart** into a C **code.**

Part I
Write each of the following a C-code form, and show the output:
1)
```
int a=15;
int *b = &a;
printf("%x %x %d\n", b, &b, *b);
```
2)
```
int *p,
int list[]={1,2,3,4};
 p = list; /* equivalent to p = &list[0] */
 printf("%d\n", *p);
```
3)
```
int *p;
 int list[]={1,2,3,4};
p = list;
printf("%d\n "p [2])
```
4)
```
int list[] = {1, 2, 3, 4} ;
printf("%d\n", list[2]);
printf(" %d\n ", *(list+2));
```
5)
```
int list[] = {1, 2, 3, 4};
int *p = list; /* same as p = &list[0] */
printf("%x",p);        // prints the address
```
6)
```
int list[] = {1, 2, 3, 4};
int *p = list;
printf( %" x",p);  // prints address
 p = p + 1;
printf("%x",p); // p must be increased by 4
```

7)
```
// sizeof() operator returns the number of needed to store a
variable or a data type
int i;
```

```
int *ptr4i = &i;
int IntArray[] = {1, 2, 3, 4, 5};
double j;
double *ptr4j = &j;
double doubleArray[] = {1.0, 2.0, 3.0, 4.0,5.0};
printf("Sizeof integer is %d bytes\n", sizeof(int));
printf("Sizeof double is %d bytes\n", sizeof(double));
printf("Sizeof i is %d bytes\n", sizeof(i));
printf("Sizeof pointer for i is %d bytes\n", sizeof(ptr4i));
printf("Sizeof j is %d bytes\n", sizeof(j));
printf("Sizeof pointer for j is %d bytes\n", sizeof(ptr4j));
printf("Sizeof intArray is %d bytes\n", sizeof(intArray));
printf("Sizeof doubleArray is %d bytes\n", sizeof(doubleArray));
```

Part II

1.

Given the following initializations:

```
int scores[] = {88, 98, 25, 53, 70, 66};
int *pscores = scores;
```

Write down the values of the following expressions:

a) *pscores ……..

b) *(pscores+2) ……..

c) pscores – scores ……

d) (pscores[2] + 5) ……

e) --*pscores ……

2. Trace

```
#include <stdio.h>
int main()
{   char *p = "ABC";
    printf("%c\n",  *(p + *p – 'B' + 2));
   return 0;}
```

3. Trace

```
#include <stdio.h>
int findx(int nm[]);

int main()
{
```

```
int k;
int numbers[5]={4 , 7 , 5 , 1 , 9} ;
char names[5]={'C','A','O','M','L'};

k = findx(numbers);
printf("First Arg = %d   Second Arg = %c",numbers[k], names[k]);
return 0;
}
int findx(int nm[])
{
int save , ps , i;
save = nm[0];
ps = 0;
for (i=1; i<5 ; i++)
    if (nm[i] < save)
   { save = nm[i];
      ps = i;
   }
printf("\n Save value = %d and coordinate = %d ",save, ps);
return ps;
}
```

4)
Write a C code to do the following task: Read a series of characters from the standard input and write them to the standard output with the characters reversed, i.e., if the input is *Ahmet* then the output will be *temhA.*

Part III

Some string functions defined in <string.h>

strlen (s1):Computes the length of the string s1, and returns the number of characters that precede '\0'.
strcat(s1 , s2): Concatenates a copy of string s2 to the end of the string s1;
strcpy(s1, s2):Copies the string s2 to s1.
strcmp(s1,s2):Compares the string s1 to the string s2. It returns a negative integer value if s1 is lexicographically less than s2, zero if s1 is equal to s2, and a positive value if s1 is lexicographically greater than s2.
strchr(s1, c) Locates the first occurrence of character c in the string s1, and returns a pointer located character if the search succeeds and NULL otherwise.

1.
Given the following initializations

```
char word1[13] = "introduction";
char word2[14] = "toprogramming";
char str1[9] = "computer";
char str2[12] = "engineering"
```

What will be the values of the following expressions ?

a) `printf("%d", strlen(word2));` ……..……..

b) `printf("%s", strncat(word1, word2, 2));` ……....

c) `printf("%s", strncpy(word2, "algorithms", 9));` ……

d) `printf("%d", strcmp(str1, str2));` ……...……..

e) `printf("%s", strcpy(str1, str2));` …….……

2.

Trace and list the output of the given program after **entering your name and surname** as input of data string from the monitor(**gets(s)**).

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

int test(void);

int i=0 ,cnt=0;
char s[20] ;
char letter[6]={"AEIOU"};

int main(void)
{
printf("ENTER Your Name  and Surname, using CAPITAL letters
\n");
gets(s);

do
 if (test()) printf("%c",s[i]);
while(s[++i]!='\0');

printf("\n Total character = %d",cnt);
return 0;
}

int test(void)
{int j;
 for(j=0;j<strlen(letter);j++)
  if (s[i]==letter[j])
```

```
            {++cnt;
                return 0;
            }
 return 1;
 }
```

3.
Write a code that reads name and surname of a person from the keyboard. Then, if the name is lexicographically greater than the surname, it prints the name first and then the surname on the monitor. Otherwise, it prints the surname first and then the name. Note that name and surname cannot be more than 30 characters each and they are assumed to be typed in lowercase characters.

4.
Write a C program for the following string of operation. Read a string of data from the monitor and after calling **split** function , divide the given string into two parts as **first** and **last** and returns back to the main program.

Split function searches '*' character in the given string and copies all the characters before '* character into **first** and copies all the remaining characters after **'*'** into **last.** Afterwards compare **first** and **last** alphabetically in the main program and display the result as follow.

First is greater than last
First is less than last
First is equal to last

Examples

Input: **book*abacus**
Output: alphabetically **book** is greater than **abacus**
Input: abdullah*adem
Output: alphabetically **abdullah** is less than **adem**
Input: deniz*deniz
Output: alphabetically **deniz** is equal to **deniz**