**PSG COLLEGE OF TECHNOLOGY, COIMBATORE – 641 004**

**DEPARTMENT OF APPLIED MATHEMATICS AND COMPUTATIONAL SCIENCES**

**Worksheet -8**

**Class : M.Sc (SS/TCS) Semester:1 Course :C Programming Lab**

1. Write and test a recursive function that returns the value of the following recursive definition:

*f*( *x* ) = 0 if *x <=* 0

*f*( *x* ) = *f* ( *x-*1) + 2 otherwise

2. Write a function that accepts an 8 by 8 array of characters that represents a maze.Each position can contain either an X or a blank. Starting at position (0,1), list anypath through the maze to get to location (7,7). Only horizontal and vertical movesare allowed. If no path exists, write a message indicating there is no path.Moves can be made only to locations that contain a blank. If an X isencountered, that path is blocked and another must be chosen. Use recursion.

3. Write a short C program that declares and initializes (to any value you like) a double, an int, and a char. Next declare and initialize a pointer to each of the three variables. Your program should then print the address of, and value stored in, and the memory size (in bytes) of each of the six variables.

4. Find out (add code to print out) the address of the variable x in foo1, and the variable y in foo2. What do you notice? Explain.

#include <stdio.h>

void foo1(intxval)

{

int x;

x = xval;

/\* print the address and value of x here \*/

}

void foo2(int dummy)

{

int y;

/\* print the address and value of y here \*/

}

int main()

{

foo1(7);

foo2(11);

return 0;

}

5. The program below uses pointer arithmetic to determine the size of a 'char' variable. By using pointer arithmetic we can find out the value of 'cp' and the value of 'cp+1'. Since cp is a pointer, this addition involves pointer arithmetic: adding one to a pointer makes the pointer point to the next element of the same type. For a pointer to a char, adding 1 really just means adding 1 to the address, but this is only because each char is 1 byte.

1. Compile and run the program and see what it does.

2. Write some code that does pointer arithmetic with a pointer to an int and determine how big an int is.

3. Same idea – figure out how big a double is, by using pointer arithmetic and printing out the value of the pointer before and after adding 1.

4. What should happen if you added 2 to the pointers from exercises 1 through 3, instead of 1? Use your program to verify your answer.

Check the other arithmetic operations like -, \*, / by int / float or address

#include <stdio.h>

int main( )

{

char c = ‘Z’;

char \*cp = &c;

printf("cp is 0x%08x\n", cp);

printf("The character at cp is %c\n", \*cp);

/\* Pointer arithmetic - see what cp+1 is \*/

cp = cp+1;

printf("cp is 0x%08x\n", cp);

/\* Do not print \*cp, because it points to

memory not allocated to your program \*/

return 0;

}

6. Write a program in C to show the basic declaration of pointer.  
*Expected Output* :

Pointer : Show the basic declaration of pointer :

-------------------------------------------------------

Here is m=10, n and o are two integer variable and \*z is an integer

z stores the address of m = 0x7ffd40630d44

\*z stores the value of m = 10

&m is the address of m = 0x7ffd40630d44

&n stores the address of n = 0x7ffd40630d48

&o stores the address of o = 0x7ffd40630d4c

&z stores the address of z = 0x7ffd40630d50

7. Write a program in C to demonstrate how to handle the pointers in the program.   
*Expected Output* :

Address of m : 0x7ffcc3ad291c

Value of m : 29

Now ab is assigned with the address of m.

Address of pointer ab : 0x7ffcc3ad291c

Content of pointer ab : 29

The value of m assigned to 34 now.

Address of pointer ab : 0x7ffcc3ad291c

Content of pointer ab : 34

The pointer variable ab is assigned with the value 7 now.

Address of m : 0x7ffcc3ad291c

Value of m : 7

8. Write a program in C to demonstrate the use of &(address of) and \*(value at address) operator.   
*Expected Output* :

Pointer : Demonstrate the use of & and \* operator :

--------------------------------------------------------

m = 300

fx = 300.600006

cht = z

Using &operator :

-----------------------

address of m = 0x7ffda2eeeec8

address of fx = 0x7ffda2eeeecc

address of cht = 0x7ffda2eeeec7

Using & and \* operator :

-----------------------------

value at address of m = 300

value at address of fx = 300.600006

value at address of cht = z

Using only pointer variable :

----------------------------------

address of m = 0x7ffda2eeeec8

address of fx = 0x7ffda2eeeecc

address of cht = 0x7ffda2eeeec7

Using only pointer operator :

----------------------------------

value at address of m = 300

value at address of fx= 300.600006

value at address of cht= z

9. Write a program in C to add two numbers using pointers.   
Test Data :  
Input the first number : 5   
Input the second number : 6   
*Expected Output* :

The sum of the entered numbers is : 11

10. Write a program in C to add numbers using call by reference.   
Test Data :  
Input the first number : 5   
Input the second number : 6   
*Expected Output* :

The sum of 5 and 6 is 11

11. Write a program in C to find the maximum number between two numbers using a pointer.

Test Data :  
Input the first number : 5   
Input the second number : 6   
*Expected Output* :

6 is the maximum number.

12. Write a program in C to store n elements in an array and print the elements using pointer.

Test Data :   
Input the number of elements to store in the array :5   
Input 5 number of elements in the array :   
element - 0 : 5   
element - 1 : 7   
element - 2 : 2   
element - 3 : 9   
element - 4 : 8   
*Expected Output* :

The elements you entered are :

element - 0 : 5

element - 1 : 7

element - 2 : 2

element - 3 : 9

element - 4 : 8

13. Write a short C program that declares and initializes (to any value you like) a double, an int, and a char. Next declare and initialize a pointer to each of the three variables. Your program should then print the address of, and value stored in, and the memory size (in bytes) of each of the six variables.

14. Find out (add code to print out) the address of the variable x in foo1, and the variable y in foo2. What do you notice? Explain.

#include <stdio.h>

void foo1(intxval)

{

intx;

x = xval;

/\* print the address and value of x here \*/

}

void foo2(int dummy)

{

inty;

/\* print the address and value of y here \*/

}

int main()

{

foo1(7);

foo2(11);

return 0;

}

15. The program below uses pointer arithmetic to determine the size of a 'char' variable. By using pointer arithmetic we can find out the value of 'cp' and the value of 'cp+1'. Since cp is a pointer, this addition involves pointer arithmetic: adding one to a pointer makes the pointer point to the next element of the same type. For a pointer to a char, adding 1 really just means adding 1 to the address, but this is only because each char is 1 byte.

1. Compile and run the program and see what it does.

2. Write some code that does pointer arithmetic with a pointer to an int and determine how big an int is.

3. Same idea – figure out how big a double is, by using pointer arithmetic and printing out the value of the pointer before and after adding 1.

4. What should happen if you added 2 to the pointers from exercises 1 through 3, instead of 1? Use your program to verify your answer.

Check the other arithmetic operations like -, \*, / by int / float or address

#include <stdio.h>

int main( )

{

char c = ‘Z’;

char \*cp = &c;

printf("cp is 0x%08x\n", cp);

printf("The character at cp is %c\n", \*cp);

/\* Pointer arithmetic - see what cp+1 is \*/

cp = cp+1;

printf("cp is 0x%08x\n", cp);

/\* Do not print \*cp, because it points to

memory not allocated to your program \*/

return 0;

}

16. swap\_nums seems to work, but not swap\_pointers. Fix it.

#include <stdio.h>

voidswap\_nums(int \*x, int \*y)

{

inttmp;

tmp = \*x;

\*x = \*y;

\*y = tmp;

}

voidswap\_pointers(char \*x, char \*y)

{

char \*tmp;

tmp = x;

x = y;

y = tmp;

}

int main()

{

inta,b;

char \*s1,\*s2;

a = 3; b=4;

swap\_nums(&a,&b);

printf("a is %d\n", a);

printf("b is %d\n", b);

s1 = "I should print second";

s2 = "I should print first";

swap\_pointers(s1,s2);

printf("s1 is %s\n", s1);

printf("s2 is %s\n", s2);

return 0;

}

17. Find and eliminate all code that generates Segmentation Fault \*/

#include <stdio.h>

int main() {

char \*\*s;

char foo[] = "Hello World";

\*s = foo;

printf("s is %s\n",s);

s[0] = foo;

printf("s[0] is %s\n",s[0]);

return(0);

}