**Short Questions:**

1. Write a single Unix command that creates an alias named **LL** to show long listing of files in your directory.

**Alias ll=’ls -al’**

1. Somewhere in some subdirectory of your current directory is a file named **tester.c**. Write the Unix command that will locate occurrences of all files with this name.

**Find . -name ‘tester.c’**

1. The present current directory on a Unix system is: **/usr/lib/X11**; the home directory is **/home/bclark**. For each of the following **cd** commands, show what would be printed by **pwd** after the command. Assume each command was run one after another.

cd ../../..  **/**

cd ~/input **/home/bclark/input**

cd – **/**

cd **/home/bclark**

1. What will be printed by the following program?

#include <stdio.h>

#define SIZE 10

int main(){

int i;

int array[SIZE];

for (i = 0; i < SIZE; i++)

array[i] = i;

int \*ptr = &array[3];

ptr++;

printf("%d ", ptr[0]);

putchar('\n');

return 0;

}

**4**

1. Write a command that shows the lines with line numbers the string “**bird**” exists in a file named “**The Rhyme of Ancient Mariner**” in your current directory.

**grep -n “bird” ‘The Rhyme of Ancient Mariner’**

1. Explain the following pointer declaration.

int (\*ptr)[10][20[30];

**This line creates a pointer to a 3D array with 10 rows, 20 columns, and 30 depth**

1. What will be printed by the following program?

#include <stdio.h>

#include <string.h>

int main(){

char expression[] = "Hello World!!!";

char name[20];

printf("Expression is: %s\n", expression);

printf("Length of expression is: %zu\n", strlen(expression));

expression[5] = '\0';

strcpy(name, expression);

printf("Name is: %s\n", name);

printf("Length of name is: %zu\n", strlen(name));

return 0;

}

**Expression is Hello World!!!**

**Length of expression is: 14**

**Name is: Hello**

**Length of name is: 5**

1. For the following array, find the number of elements using **sizeof** operator.

int y[][4] = { {2,1,4,5}, {12, 14, 16, 18}, {9, 3, 2, 1} };

**sizeof(y)/sizeof(int) = 12 integers**

1. Explain **&i**, **ptr**, **\*ptr**, and **&ptr** in the context of the following code snippet.

int i = 10;

int \*ptr = &i;

**&I is the memory address of integer i**

**Ptr is a pointer containing the memory address of integer i**

**\*ptr is the dereferenced integer i, equal to the value of integer i**

**&ptr is the memory of address of pointer ptr itself**

1. Explain the following Unix command.

grep -n string \*.[ch]

**find all occurrences, with line numbers, of the string “string” in all .c or .h files in the current directory**

1. In the following code, complete ‘**updateElement**’ function so that it adds **10** to a particular array element that is passed as a function argument as shown in the line marked with an arrow.

#include <stdio.h>

#define SIZE 5

void updateElement(int \*p);

int main(){

int a[SIZE] = { 0, 1, 2, 3, 4 };

int i;

printf("The values of elements in the original array are:\n");

for (i = 0; i < SIZE; i++){

printf(" %d", a[i]);

}

printf("\n");

**updateElement(&a[2]);** 🡨

printf("The values in the modified array are:\n");

for (i = 0; i < SIZE; i++){

printf(" %d", a[i]);

}

return 0;

}

void updateElement(int\* element){

**\*element += 10;**

}

1. Explain the following two expressions.

int (\*p)[10];

int \*ptr[10];

**The first line creates a pointer to an array of 10 integers.**

**The second line creates an array of 10 integer pointers.**

1. . What will be printed by the following program?

#include <stdio.h>

int main(){

int (\*ptr)[10];

int a[10]={99,1,2,3,4,5,6,7,8,9};

ptr= &a;

printf("%d", \*\*ptr);

return 0;

}

**99**

1. Explain the following two expressions.

double \*p1();

double (\*p1)();

**The first line declares a function p1 that takes 0 arguments and returns a double pointer.**

**The second line creates a pointer to a function p1 that takes 0 arguments and returns a double**

1. What will be printed by the following program?

#include <stdio.h>

void function(int array[]) {

printf("size = %zu\n", sizeof(array));

printf("x[4] = %d\n", array[4]);

array = array + 1;

printf("x[4] = %d\n", array[4]);

}

int main(void) {

int x[] = { 1, 2, 3, 4, 5, 6, 7, 8, 9 };

printf("size = %zu\n", sizeof(x));

printf("x[4] = %d\n", x[4]);

function(x);

return 0;

}

**Size = 36**

**X[4] = 5**

**Size = 8**

**X[4] = 5**

**X[4] = 6**

**Identify whether the following statements are True/False.**

1. Memory allocation function **realloc()** can only grow the memory block granted previously by a successful call to **malloc()**, **calloc()**,or **realloc()**.
   * **false**
2. While moving through an array, you must use indexing while you can simply increment a pointer.
   * **true**
3. The metacharacter **“^”** is used in regular expressions to match a particular string at the beginning of a line.
   * **true**
4. A relative path is a path that starts from the home directory.
   * **false**
5. A null character (‘\0’) is appended to the end of the string when a sentence is read from **stdin** using **fgets()**.
   * **true**
6. Arrays are stored in memory in row-major order.
   * **true**
7. Void pointers can be freely assigned to any type of pointer without an explicit cast being required.
   * **true**
8. ‘**grep**’ command is recursive by default.
   * **false**
9. The following command will redirect the standard output for the listing of all .c files to output.txt.

ls –l \*.c 2> output.txt

**false**

1. Considering the following expressions where ‘**ptr**’ is a constant pointer, the expression in the line with an arrow is correct.

char c[10];

char \*const ptr = &c[0];

ptr = &c[3]; 🡨

**false**

1. The following two expressions are equivalent:

**\*(\*(x+2)+5)** and **x[2][5]**

**True, if x is a pointer**

12. In the expression below, ‘**p**’ is initialized with the address of the first row in ‘**y**’.

int y[4][5];

int (\*p)[5] = y;

**true**

13. A pointer to a pointer points to the address of another pointer.

**true**

14. Consider an array declared as follows: int x[10] = {0, 1, 2, 3, 4, 5, 6, 7, 8, 9};

The following declaration is correct:

printf(“Value of element at index ‘1’ is: %d\n”, 1[x]);

**false**

15. The following Unix command will print “Hello Joseph”.

avar=Joseph

echo Hello $avar

**true**