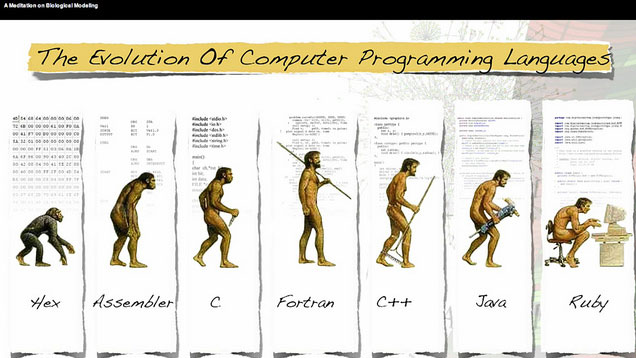
***COP2270***

***M/W***

***Spring 2017-2018***



***Professor: Yassin Raef***

***Anaisy Garcia***

***Chapter 5 Homework/ Exercises***

***Duplicate Figures 5.3, 5.4, 5.6, 5.11, 5.12, 5.13, 5.14,5.16, 5.18,5.19***

***Do Problem(s) 5.15, 5.16, 5.20, 5.25, 5.31, 5.41***

***One problem per page Please***

***Figure 5-3***

|  |
| --- |
| ***Commands*** |
| ***// Fig. 5.3: fig05\_03.c***  ***// Creating and using a programmer-defined function.***  ***#include <stdio.h>***  ***int square(int y); // function prototype***  ***int main(void)***  ***{***  ***// loop 10 times and calculate and output square of x each time***  ***for (int x = 1; x <= 10; ++x) {***  ***printf("%d ", square(x)); // function call***  ***}***  ***puts("");***  ***}***  ***// square function definition returns the square of its parameter***  ***int square(int y) // y is a copy of the argument to the function***  ***{***  ***return y \* y; // returns the square of y as an int***  ***}*** |

|  |
| --- |
| ***Output*** |
|  |

***Figure 5-4***

|  |
| --- |
| ***Commands*** |
| ***// Fig. 5.4: fig05\_04.c***  ***// Finding the maximum of three integers.***  ***#include <stdio.h>***  ***int maximum(int x, int y, int z); // finction prototype***  ***int main (void)***  ***{***  ***int number1;***  ***int number2;***  ***int number3;***  ***printf("%s", "Enter three integers: ");***  ***scanf("%d%d%d", &number1, &number2, &number3);***  ***//number1, number2, and number3 are arguments***  ***//to the maximum function call***  ***printf("Maximum is: %d\n", maximum(number1, number2, number3));***  ***}***  ***// Function maximum definition***  ***// x, y, and z are parameters***  ***int maximum(int x, int y, int z)***  ***{***  ***int max = x;***  ***if(y > max) {***  ***max = y;***  ***}***  ***if(z > max) {***  ***max = z;***  ***}***  ***return max;***  ***}*** |

|  |
| --- |
| ***Output*** |
|  |

***Figure 5-6***

|  |
| --- |
| ***Commands*** |
| ***// Fig. 5.6: fig05\_06.c***  ***// Demonstrating the function call stack***  ***// and stack frames using a funtion square.***  ***#include <stdio.h>***  ***int square(int); // prototype for function square***  ***int main()***  ***{***  ***int a = 10; // value to square (local automatic variable in main)***  ***printf("%d squared: %d\n", a, square(a)); // display a squared***  ***}***  ***// returns the square of an integer***  ***int square(int x) // x is a local variable***  ***{***  ***return x \* x; // calculate square and return result***  ***}*** |

|  |
| --- |
| ***Output*** |
|  |

***Figure 5-11***

|  |
| --- |
| ***Commands*** |
| ***// Fig. 5.11: fig05\_11.c***  ***// Shifted, scaled random integers produced by 1 + rand() % 6.***  ***#include <stdio.h>***  ***#include <stdlib.h>***  ***int main(void)***  ***{***  ***// loop 20 times***  ***for (unsigned int i = 1; i <= 20; ++i) {***  ***// pick random number from 1 to 6 and output it***  ***printf("%10d", 1 + (rand() % 6));***  ***// if counter is divisible by 5, begin new line of output***  ***if (i % 5 == 0) {***  ***puts("");***  ***}***  ***}***  ***}*** |

|  |
| --- |
| ***Output*** |
|  |

***Figure 5-12***

|  |
| --- |
| ***Commands*** |
| ***// Fig. 5.12: fig05\_12.c***  ***// Rolling a six-headed die 60,000,000 times.***  ***#include <stdio.h>***  ***#include<stdlib.h>***  ***int main (void)***  ***{***  ***unsigned int frequency1 = 0; // rolled 1 counter***  ***unsigned int frequency2 = 0; // rolled 2 counter***  ***unsigned int frequency3 = 0; // rolled 3 counter***  ***unsigned int frequency4 = 0; // rolled 4 counter***  ***unsigned int frequency5 = 0; // rolled 5 counter***  ***unsigned int frequency6 = 0; // rolled 6 counter***  ***// loop 60000000 times and summarize results***  ***for (unsigned int roll = 1; roll <= 60000000; ++roll) {***  ***int face = 1 + rand() % 6; // random number from 1 to 6***  ***// determine face value and increment appropriate counter***  ***switch (face) {***  ***case 1: // rolled 1***  ***++frequency1;***  ***break;***  ***case 2: // rolled 1***  ***++frequency2;***  ***break;***  ***case 3: // rolled 1***  ***++frequency3;***  ***break;***  ***case 4: // rolled 1***  ***++frequency4;***  ***break;***  ***case 5: // rolled 1***  ***++frequency5;***  ***break;***  ***case 6: // rolled 1***  ***++frequency6;***  ***break; // optional***  ***}***  ***}***  ***// display results in tabular format***  ***printf("%s%13s\n", "Face", "Frequency");***  ***printf(" 1%13u\n", frequency1);***  ***printf(" 2%13u\n", frequency2);***  ***printf(" 3%13u\n", frequency3);***  ***printf(" 4%13u\n", frequency4);***  ***printf(" 5%13u\n", frequency5);***  ***printf(" 6%13u\n", frequency6);***  ***}*** |

|  |
| --- |
| ***Output*** |
|  |

***Figure 5-13***

|  |
| --- |
| ***Commands*** |
| ***// Fig. 5.13: fig05\_13.c***  ***// Randomizing the die-rolling program.***  ***#include <stdlib.h>***  ***#include <stdio.h>***  ***int main(void)***  ***{***  ***unsigned int seed; // number used to seed the random number generator***  ***printf("%s", "Enter seed: ");***  ***scanf("%u", &seed); // note %u for unsigned int***  ***srand(seed); // seed the random number generator***  ***// loop 10 times***  ***for(unsigned int i = 1; i <= 10; ++i) {***  ***// pick a random number from 1 to 6 and output it***  ***printf("%10d", 1 + (rand() % 6));***  ***// if counter is divisible by 5, begin a new line of output***  ***if(i % 5 == 0) {***  ***puts("");***  ***}***  ***}***  ***}*** |

|  |
| --- |
| ***Output*** |
|  |

***Figure 5-14***

|  |
| --- |
| ***Commands*** |
| ***// Fig. 5.14: fig05\_14.c***  ***// Simulating the game of craps.***  ***#include <stdio.h>***  ***#include <stdlib.h>***  ***#include <time.h> // contains prototype for funtion time***  ***// enumeration constants represent game status***  ***enum Status { CONTINUE, WON, LOST};***  ***int rollDice(void); // function prototype***  ***int main(void)***  ***{***  ***// randomizze random number generator using current time***  ***srand(time(NULL));***  ***int myPoint; // player must make this point to win***  ***enum Status gameStatus; // can contain CONTINUE, WON, or LOST***  ***int sum = rollDice(); // first roll of the dice***  ***// determine game status based on sum of dice***  ***switch(sum) {***  ***// win on first roll***  ***case 7: // 7 is a winner***  ***case 11: // 11 is a winner***  ***gameStatus = WON;***  ***break;***  ***// lose on first roll***  ***case 2: // 2 is a loser***  ***case 3: // 3 is a loser***  ***case 12: // 12 is a loser***  ***gameStatus = LOST;***  ***break;***  ***// remember point***  ***default:***  ***gameStatus = CONTINUE; // player should keep rolling***  ***myPoint = sum; // remember the point***  ***printf("Point is %d\n", myPoint);***  ***break; // optional***  ***}***  ***// while game not complete***  ***while (CONTINUE == gameStatus) { // player should keep rolling***  ***sum = rollDice(); // roll dice again***  ***// determine game status***  ***if (sum == myPoint) { // win by making point***  ***gameStatus = WON;***  ***}***  ***else {***  ***if (7 == sum) { // lose by rolling 7***  ***gameStatus = LOST;***  ***}***  ***}***  ***}***  ***// display won or lost message***  ***if (WON == gameStatus) { // did player win?***  ***puts("Player wins");***  ***}***  ***else {***  ***puts("Player loses");***  ***}***  ***}***  ***// roll dice, calculate sum and display results***  ***int rollDice(void)***  ***{***  ***int die1 = 1 + (rand() % 6); // pick random die1 value***  ***int die2 = 1 + (rand() % 6); // pick random die2 value***  ***// display results of this roll***  ***printf("Player rolled %d + %d = %d\n", die1, die2, die1 + die2);***  ***return die1 + die2; // return sum of dice***  ***}*** |

|  |
| --- |
| ***Output*** |
|  |

***Figure 5-16***

|  |
| --- |
| ***Commands*** |
| ***// Fig. 5.16: fig05\_16.c***  ***// Scoping.***  ***#include <stdio.h>***  ***void useLocal(void); // function prototype***  ***void useStaticLocal(void); // function prototype***  ***void useGlobal(void); // function prototype***  ***int x = 1; // global variable***  ***int main(void)***  ***{***  ***int x = 5; // local variable to main***  ***printf("local x in ouer scope of main is %d\n", x);***  ***{ // start new scope***  ***int x =7; // local variable to new scope***  ***printf("local x in inner scope of main is %d\n", x);***  ***} // end new scope***  ***printf("local x in outer scope of main is %d\n", x);***  ***useLocal(); // useLocal has automatic local x***  ***useStaticLocal(); // useStaticLocal has static local x***  ***useGlobal(); // useGlobal uses global x***  ***useLocal(); // useLocal reinitializes automatic local x***  ***useStaticLocal(); // static local x retains its prior value***  ***useGlobal(); // global x also retains its value***  ***printf("\nlocal x in main is %d\n", x);***  ***}***  ***// useLocal reinitializes local variable x during each call***  ***void useLocal(void)***  ***{***  ***int x = 25; // initialized each time useLocal is called***  ***printf("\nlocal x in useLocal is %d after entering useLocal\n", x);***  ***++x;***  ***printf("local x in useLocal is %d before exiting useLocal\n", x);***  ***}***  ***// useStaticLocal initializes static local variable x only the first time***  ***// the function is called; value of x is saved between calls to this***  ***// function***  ***void useStaticLocal(void)***  ***{***  ***// initialized once***  ***static int x = 50;***  ***printf("\nlocal static x is %d on entering useStaticLocal\n", x);***  ***++x;***  ***printf("local static x is %d on exiting useStaticLocal\n", x);***  ***}***  ***// function useGlobal modifies global variable x during each call***  ***void useGlobal(void)***  ***{***  ***printf("\nglobal x is %d on entering useStaticLocal\n", x);***  ***x \*= 10;***  ***printf("global x is %d on exiting useStaticLocal\n", x);***  ***}*** |

|  |
| --- |
| ***Output*** |
|  |

***Figure 5-18***

|  |
| --- |
| ***Commands*** |
| ***// Fig. 5.18: fig05\_18.c***  ***// Recursive factorial function.***  ***#include <stdio.h>***  ***unsigned long long int factorial(unsigned int number);***  ***int main(void)***  ***{***  ***// during each iteration, calculate***  ***// factoral(i) and display result***  ***for (unsigned int i = 0; i <= 21; ++i){***  ***printf("%u! = %llu\n", i, factorial(i));***  ***}***  ***}***  ***// recursive definition of function factorial***  ***unsigned long long int factorial(unsigned int number)***  ***{***  ***// base case***  ***if (number <= 1) {***  ***return 1;***  ***}***  ***else { // recursive step***  ***return (number \* factorial(number - 1));***  ***}***  ***}*** |

|  |
| --- |
| ***Output*** |
|  |

***Figure 5-19***

|  |
| --- |
| ***Commands*** |
| ***// Fig. 5.19: fig05\_19.c***  ***// Recursive fibonacci function***  ***#include <stdio.h>***  ***unsigned long long int fibonacci(unsigned int n); // function prototype***  ***int main(void)***  ***{***  ***unsigned int number; // number input by user***  ***// obtain integer from user***  ***printf("%s", "Enter an integer: ");***  ***scanf("%u", &number);***  ***// calculate fibonacci value for number input by user***  ***unsigned long long int result = fibonacci(number);***  ***//display result***  ***printf("Fibonacci(%u) = %llu\n", number, result);***  ***}***  ***// Recursive definition of funtion fibonacci***  ***unsigned long long int fibonacci(unsigned int n)***  ***{***  ***// base case***  ***if (0 == n || 1 == n) {***  ***return n;***  ***}***  ***else { // recursive step***  ***return fibonacci(n - 1) + fibonacci(n - 2);***  ***}***  ***}*** |

|  |
| --- |
| ***Output*** |
|  |

***Figure 5-15***

|  |
| --- |
| ***Commands*** |
| // Fig5.15 #include <stdio.h> #include <math.h>  double hypotenuse(double firstside, double secondside);  int main() { int a; double firstside; double secondside;  for(a = 1; a <= 3; a++) { printf("Enter first side of triangle known: "); scanf("%lf", &firstside); printf("Enter second side of triangle known: "); scanf("%lf", &secondside); printf("Hypotenuse: %.1f\n\n", hypotenuse(firstside, secondside)); } // end for return 0;  // end main } double hypotenuse (double firstside, double secondside) {     return sqrt( pow(firstside, 2) + pow(secondside, 2)); } // end double |

|  |
| --- |
| ***Output*** |
|  |

***Figure 5-16***

|  |
| --- |
| ***Commands*** |
| // Fig 5.16 #include <stdio.h>  int integerPower (int b, int e);  int main() { int exponent; int base;  printf("Enter base: "); scanf("%d", &base); printf("Enter exponent: "); scanf("%d", &exponent); printf("%d to the power %d is: %d\n", base, exponent, integerPower(base, exponent));  return 0; } // end main  int integerPower(int b, int e) { int product = 1; int x;  for (x = 1; x <= e; x++) { product \*= b; }  return product; } |

|  |
| --- |
| ***Output*** |
|  |

***Figure 5-20***

|  |
| --- |
| ***Commands*** |
|  |

|  |
| --- |
| ***Output*** |
|  |

***Figure 5-25***

|  |
| --- |
| ***Commands*** |
|  |

|  |
| --- |
| ***Output*** |
|  |

***Figure 5-31***

|  |
| --- |
| ***Commands*** |
| ***// Fig5.31***  ***#include <stdio.h>***  ***#include <stdlib.h>***  ***#include <time.h>***  ***int flip ();***  ***int main(void)***  ***{***  ***unsigned int loop;***  ***int headCount = 0;***  ***int tailCount = 0;***  ***srand(time(NULL));***  ***for(loop =1; loop <= 100; ++loop) {***  ***if(flip() == 0) {***  ***++tailCount;***  ***} // and if***  ***else {***  ***++headCount;***  ***} // end else***  ***if(loop % 10 == 0) {***  ***puts("");***  ***} // end second if***  ***} // end for***  ***printf("\nThe total number of Heads was %d\n", headCount);***  ***printf("The total number of Tails was %d\n", tailCount);***  ***} // end main***  ***int flip()***  ***{***  ***int HorT = (rand() %2);***  ***if(HorT == 0) {***  ***printf("%s", "Tails");***  ***} // end flip if***  ***else {***  ***printf("%s", "Heads");***  ***} // end flip else***  ***return HorT;***  ***}*** |

|  |
| --- |
| ***Output*** |
|  |

***Figure 5-41***

|  |
| --- |
| ***Commands*** |
|  |

|  |
| --- |
| ***Output*** |
|  |