1. //Precedence
3. **void** precedence()
4. {
5. **int** a = 2;
7. //This expression is invalid cause 3<a is a boolean which returns 0
8. //then 0 ==0 returns 1 cause its true
9. printf("%d\n", (3<a==0));
10. //From this we can see associativity is left to right.
11. **return** 0;
12. }
13. //Ordinal
15. **void** printOrdinal (**int** year) {
16. **int** century = year/100 + 1;
18. **int** suffix = century%100;
20. **if** (suffix < 14 && suffix > 10) {
21. printf("th");
22. }**else**{
23. **switch**(century%10){
24. **case** 1:
25. printf("st");
26. **break**;
28. **case** 2:
29. printf("nd");
30. **break**;
32. **case** 3:
33. printf("rd");
34. **break**;
36. **default**:
37. printf("th");
38. }
39. }
40. printf(" century.\n");
41. }
42. //Coin test
44. **int** main() {
45. **int** i=0, amount, coins;
46. **int** coinArray[6] = {100, 50, 20, 10, 5, 1};
48. printf("enter amount in cents \n");
49. scanf("%d", &amount);

52. **while** (amount > 0) {
53. **if** (amount >= coinArray[i]) {
54. amount -= coinArray[i];
55. coins++;
56. }**else**{
57. i++;
58. }
59. }
60. printf("%d \n", coins);
62. **return** 0;
63. }
64. //Euclid algorithm
66. **int** euclid(**int** a, **int** b) {
67. **int** r;
68. **while** (b>0) {
69. r = a%b;
70. a = b;
71. b=r;
72. }
74. **return** a;
75. }
76. //Prime checker
78. **int** is\_prime(**int** n) {
79. **int** counter = 2, boolean = 1;
80. **while** (counter<n) {
81. **if** (n%counter == 0) {
82. boolean = 0;
83. **break**;
85. }**else**{
86. counter++;
87. }
88. }
90. **if** (n == 1 || n == 0 ){
91. boolean = 0;
92. }
93. **return** boolean;
94. }
95. //Number splitter
96. **void** disectNum(**int** inputNum) {
97. **while**(inputNum){
98. printf("%d\n", inputNum%10);
99. inputNum /= 10;
100. }
101. }
102. //first 2 numbers
104. **void** first2Numbers (**int** first2) {
106. **while** (first2/count >= 100) {
107. count \*= 10;
108. }
110. first2 /= count;
111. }

Calculating parking fees from time in and time out

1. **double** computeWeekdays (**int** timeIn, **int** timeOut) {
2. **double** fees;
4. **int** converted\_timeIn = processTime(timeIn);
5. **int** converted\_timeOut = processTime(timeOut) + 60 - processTime(timeOut)%60;
7. **while** (converted\_timeIn < converted\_timeOut) {
8. //4am to 7pm
9. **if** (converted\_timeIn >= 240 && converted\_timeIn < 420) {
10. fees += 2.00;
11. converted\_timeIn += 60;
12. }
13. //7am to 6pm
14. **else** **if** (converted\_timeIn >= 420 && converted\_timeIn < 1080) {
15. fees += 1.20;
16. converted\_timeIn += 30;
17. }
18. //6pm to midnight
19. **else** **if** (converted\_timeIn >= 1080) {
20. fees += 5.00;
21. **break**;
22. }
23. }
25. //surcharge
26. **if** ((processTime(timeOut) - processTime(timeIn))/60 >= 10) {
27. fees \*= 1.1;
28. }
30. **if** (processTime(timeOut) > 1320) {
31. fees += 3.00;
32. }
34. **return** fees;
35. }

Calculating number of possibilities to form coins using **brute force**

1. **int** main () {
3. **int** amnt , possible = 0;
5. printf("Enter amount: ");
6. scanf("%d", &amnt);
8. **int** a = 0, b = 0, c = 0, d = 0;

11. **while** (a<=20) {
12. **int** n = amnt;
14. n -= a\*20;
15. n -= b\*10;
16. n -= c\*5;
17. n -= d\*1;
19. d++;
21. **if** (d == 21) {
22. d = 0;
23. c++;
24. }
26. **if** (c == 21) {
27. c = 0;
28. b++;
29. }
31. **if** (b == 21) {
32. b = 0;
33. a++;
34. }
36. **if** (n == 0 && amnt > 0) {
37. possible++;
38. }
39. }

42. printf("Possibilities: %d\n", possible);
44. **return** 0;
45. }

Number splitter in chunks

2. **int** main () {
3. // Splitting chunks of number
5. **int** n = 1234567890;
6. **long** **long** num = 1234567890;

9. **while** (n) {
10. printf ("Chunk: %d\n", n%1000);
12. n /= 1000;
13. }
14. // splits the digits from the bottom up
15. // OUTPUT: 890 567 234 1
17. **int** i;
18. **int** numberSegment = 3, numberOfDigits = 10;
19. **for** (i = numberOfDigits; i >= 0; i-=numberSegment) {
20. printf("Chunk top: %lli\n", (num/(**long** **long**)pow(10, i))%(**long** **long**)pow(10,numberSegment));
21. }
22. // splits the digits from the top down
23. // OUTPUT: 0 123 456 789
25. }

Binary sort for numbers:

1. **int** main () {
3. **int** high = 200000000, low = 0, guess = 3, myGuess;
5. printf("Enter number to guess: ");
6. scanf("%d", &guess);
8. **while** (myGuess != guess) {
9. myGuess = (high - low)/2 + low;
11. **if** (myGuess > guess) {
12. printf("High guess: %d\n", myGuess);
13. high = myGuess;
14. }**else** {
15. printf("Low guess: %d\n", myGuess);
16. low = myGuess;
17. }
18. }
20. printf("My guess: %d\n", myGuess);
22. }

Impromptu number rounding:

1. **int** roundUp (**int** number) {
2. **int** rounded;
3. **float** number\_f = number;
4. **if** (number\_f/someNumber - number/someNumber >= 0.5) {
5. rounded = number/someNumber + 1;
6. }**else** {
7. rounded = number/someNumber;
8. }

11. **return** rounded;
12. }