|  |  |  |  |
| --- | --- | --- | --- |
| ID (Count) | File Name | Code Block | Equivalent Mutant |
| 1. | DecimalToBinary.c | while (power > 0) {  if (num >= power) {  firstOne = 1;  printf("1");  num %= power; | Replace the num %= power  With:  num = num%power |
| 2 | IncrementOperator.c | int main()  {  int a= 5, b, c;  b = a++ + ++a; //Right to left | Replace b =a++ + ++a;  With  b = a ;  a++;  b += ++a; |
| 3 | IncrementOperator.c | int main()  {  int a= 5, b, c;  b = a++ + ++a; //Right to left  c = ++a + a++; //Right to left | Replace c= ++a + a++;  With:  a++;  c = a + a++; |
| 4 | Increment\_number.c | int main(void) {  int num = 0; // This is the Number that will be incremented.  int increment = 1; // This is what we'll be using to increment(add) onto num.    // While num's Value is less than 100, we'll increment the Value of num by 1 every second.  while(num < 100) {  sleep(1); // Wait 1 second each time, num has it's Value incremented  num += increment; // Increment num | Replace num += increment with:  num++; |
| 5 | Decrement\_operator | int main()  {  int a= 5, b, c;  b = a--;  c = a--;  printf("Value of b = %d\n Value of c = %d\n", b,c);  return 0;  } | Replace b = a—with:  b=a;  a--; |
| 6 | Find\_val.c | int findVal(int numbers[], int length, int val)  {  int findVal = -1;  int i;    for (i = 0; i < length; i++)  if (numbers[i] == val)  findVal = i;  return findVal;  } | Replace for (i = 0; i < length; i++)  if (numbers[i] == val)  findVal = i with:  for (i = lrngth-1; i >=0; i--)  if (numbers[i] == val)  return i; |
| 7 | DailyWageCalc.c | if (hour >= 1 && hour <= 8)  amount = 100;  else if (hour >= 9 && hour <= 12)  amount = 100 + (hour - 8)\*20; | Replace else if with: if |
| 8 | jday-jdate.c | if(m < 10)  m += 3;  else {  m -= 9;  ++y;  } | Replace ++y with: y++; |
| 9 | CheckPalindrome.c | while (low < high) {  if (s[low] != s[high])  return 0; /\* Not a palindrome\*/  low++;  high--; | Replace low<high with:  low != high |
| 10 | CheckPalindrome.c | while (low < high) {  if (s[low] != s[high])  return 0; /\* Not a palindrome\*/  low++;  high--; | Replace high—with:  High -=1; |
| 11 | even\_or\_not.c | if (num % 2 == 0)  // jump to even  goto even;  else  // jump to odd  goto odd; | Replace num%2 with:  num –2\*(num/2) |
| 12 | Year.c | int i;  for (i=Jan; i<=Dec; i++)  printf("%d ", i); | Replace i=Jan with: i=0 |
| 13 | Max.c | int max(int x, int y)  {  if (x > y)  return x;  else  return y;  } | Replace if (x > y)  return x;  else  return y;  with: return x>y?x:y; |
| 14 | oddOrPos.c | for (i = 0; i < length; i++) {  if (x[i] % 2 == 1 || x[i] % 2 == -1 || x[i] > 0) {  count++; | Replace x[i]%2==1|| x[i]%2==-1 with:  X[i] %2 !=0 |
| 15 | Sum.c | int sum(int x[], int size)  {  int s = 0;  int i;    for (i = 0; i < size; i++) {  s = s + x[i];  }  return s;  } | Replace s = s+x[i] with:  s+= x[i] |
| 16 | Sum.c | int sum(int x[], int size)  {  int s = 0;  int i;    for (i = 0; i < size; i++) {  s = s + x[i];  }  return s;  } | Replace i=0; i<size; i++ with:  i=size-1;i>=0;i-- |
| 17 | Sum\_one.c | int sum (int a, int b)  {  int s;  s=a+b;  return s;  } | Replace return s; with:  Return a+b; |
| 18 | Sorted\_list.c | else{  tmp = array[i];  array[i] = array[i-1];  array[i-1] = tmp;  --i;  if(i == 0)  i = 1;  } | Replace –I; if (i==0) i=1; with:  If(i>1)  –i; |
| 19 | Sorted\_list.c | for(i=1; i<size; ){  if(array[i-1] <= array[i])  ++i; | Replace array[i-1] <=array[i] with:  Array[i] > array[i-1] |
| 20 | Bogo.c | bool is\_sorted(int \*a, int n)  {  while ( --n >= 1 ) {  if ( a[n] < a[n-1] ) return false;  }  return true;  } | Replace return false; with:  return a[n]>=a[n-1]; |
| 21 | DigitalRoot.c | unsigned int number, temp, droot = 0;  printf("Enter a positive number: ");  scanf("%u", &number);  temp = number; | Replace scanf (“%u”, &number);  Temp=number; with:  Scanf(“%u”, &temp); |
| 22 | DigitalRoot.c | while(temp != 0) {  int digit = temp % 10;  droot += digit;  temp /=10;  if(temp == 0 && droot > 9) {  temp = droot;  droot = 0;  } | Replace temp!=0 with:  Temp>0 |
| 23 | BinarySearch.c | if(number == array[tmp]) {  return 0;  } else if(number > array[tmp]) {  return binarySearch(array, number, start, tmp);  } else {  return binarySearch(array, number, tmp, end);  } | Replace else if with: if |
| 24 | BinarySearch.c | if(start >= end) {  return array[start] == number ? 0 : 1;  } | Replace return array[start] == number? 0:1; with:  Return array[start] ==number; |
| 25 | DecimalToBinary.c | while (power > 0) {  if (num >= power) {  firstOne = 1;  printf("1");  num %= power;  } else if (firstOne == 1) {  printf("0");  } | Replace else if(fistOne == 1) { with: else { if(firstOne==1) |
| 26 | Diagonal-Difference.c | if(arr[i][j]>=-100 && arr[i][j]<=100)  {  if(i==j)  {  sum1+=arr[i][j];  }  if(j==(n-1-i))  {  sum2+=arr[i][j];  }  }  } | Replace arr[i][j]>=100 && arr[i][j] <=100 with:  arr[i][j]>-101 && arr[i][j]<101 |
| 27 | Diagonal-Difference.c | if((sum1-sum2)<0)  {  printf("%d", (-((sum1)-(sum2))));  }  else  {  printf("%d", ((sum1)-(sum2)));  } | Replace entire code block with:  Printf (“%d”, sum1-sum2<0? (-((sum1)-(sum2))): (-((sum1)-(sum2))); |
| 28 | Diagonal-Difference.c | if((sum1-sum2)<0)  {  printf("%d", (-((sum1)-(sum2))));  }  else  {  printf("%d", ((sum1)-(sum2)));  } | Replace sum1-sum2<0 with:  abs(sum1) < abs(sum2)&&sum1<sum2 |
| 29 | FibonacciGeneration.c | int fib(int n)  {  // Base case defined  if (n <= 1){  return n;  } | Replace n<=1 with: n<2 |
| 30 | DailyWageCalc.c | if (hour >= 1 && hour <= 8)  amount = 100;  else if (hour >= 9 && hour <= 12)  amount = 100 + (hour - 8)\*20; | Replace amount = 100 + (hour-8)\*20; with:  amount = 180 –(12-hour)\*20; |