|  |  |  |  |
| --- | --- | --- | --- |
| ID (Count) | File Name | Code Block | Equivalent Mutant |
| 1. | max.c | int max(int x, int y)  {  if (x > y)  return x;  else  return y;  } | Replace the if-else statement with:  return (x > y) ? x : y; |

|  |  |  |  |
| --- | --- | --- | --- |
| 2. | increment\_number.c | while (num < 100) {  sleep(1);  num += increment;  printf("%d\n", num);  } | Replace sleep(1) with:  usleep(1000000); |
| 3. | FibonacciGeneration.c | int fib(int n)  {  if (n <= 1){  return n;  }  return fib(n - 1) + fib(n - 2);  } | Swapping the order of recursive calls:  return fib(n - 2) + fib(n - 1); |
| 4. | DigitalRoot.c | while(temp != 0) {  int digit = temp % 10;  droot += digit;  temp /=10;  if(temp == 0 && droot > 9) {  temp = droot;  droot = 0;  }  } | Replace the statement with:  droot=droot+digit; |
| 5. | DigitalRoot.c | while(temp != 0) {  int digit = temp % 10;  droot += digit;  temp /=10;  if(temp == 0 && droot > 9) {  temp = droot;  droot = 0;  }  } | Replace the statement with:  temp=temp/10; |
| 6. | DecrementOperator.c | 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  c= a-- with:  c=a |
| 7. | DailyWageCalc.c | int main(){  int hour, amount;  printf("Enter number of duty hours\n");  scanf("%d", &hour);  if (hour >= 1 && hour <= 8)  amount = 100;  else if (hour >= 9 && hour <= 12)  amount = 100 + (hour - 8)\*20;  else if (hour >= 13 && hour <= 16)  amount = 180 + (hour - 12)\*40;  else if (hour >= 17 && hour <= 20)  amount = 340 + (hour - 16)\*60;  else if (hour >= 21 && hour <= 24)  amount = 580 + (hour - 20)\*80;  printf("Amount incurred by worker : %d", amount);  return 0;  } | adds an additional else block at the end, assigning 0 to amount when the hour is outside the valid range:  else  amount = 0; // Equivalent mutant: Assigning 0 when hour is outside valid range |
| 8. | BinarySearch.c | int binarySearch(int array[], int number, int start, int end) {  if(start >= end) {  return array[start] == number ? 0 : 1;  }  int tmp = (int) end / 2;  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 end/2 with:  (start + end) / 2  The original code incorrectly used (int) end / 2 which may truncate the value and produce incorrect results in certain cases. The mutant code fixes this issue and ensures the correct middle index is calculated during the binary search. |
| 9. | printtokens.c | unget\_char(ch,stream\_ptr)  CHARACTER ch;  character\_stream stream\_ptr;  {  if(stream\_ptr->stream\_ind == 0)  return 0;  else stream\_ptr->stream[--(stream\_ptr->stream\_ind)]=ch;  return 0;} | Change the decrement operation --(stream\_ptr->stream\_ind) to (stream\_ptr->stream\_ind - 1) to assign ch to the previous position in the stream array, and then decrement stream\_ptr->stream\_ind:  stream\_ptr->stream[stream\_ptr->stream\_ind - 1] = ch;  (stream\_ptr->stream\_ind)--; |
| 10. | shaker.c | for(it = 0; it < 2; ++it) {  flag = 1;  for(i = start[it]; i != end[it]; i += inc[it])  if(a[i - 1] > a[i]) {  swap(a + i - 1, a + i);  flag = 0;  }  if(flag)  return;} | Replace< with:  != |
| 11. | shaker.c | for(it = 0; it < 2; ++it) {  flag = 1;  for(i = start[it]; i != end[it]; i += inc[it])  if(a[i - 1] > a[i]) {  swap(a + i - 1, a + i);  flag = 0;  }  if(flag)  return;  } | Replace> with:  >= |
| 12. | bogo.c | void shuffle(int \*a, int n)  {  int i, t, r;  for(i=0; i < n; i++) {  t = a[i];  r = rand() % n;  a[i] = a[r];  a[r] = t;  }  } | Replace< with:  != |
| 13. | bogo.c | int main()  {  int x[] = { 1, 10, 9, 7, 3, 0 };  int i;  int len = sizeof(x)/sizeof(x[0]);  printf("Original Array:\n");  for (i = 0; i < len; i++)  printf("%d%s", x[i], i == len - 1 ? "\n" : " ");  printf("\nSorted Array:\n");  bogosort(x, 6);  for (i=0; i < 6; i++) printf("%d ", x[i]);  printf("\n");  } | Replace< with:  != |
| 14. | bogo.c | int main()  {  int x[] = { 1, 10, 9, 7, 3, 0 };  int i;  int len = sizeof(x)/sizeof(x[0]);  printf("Original Array:\n");  for (i = 0; i < len; i++)  printf("%d%s", x[i], i == len - 1 ? "\n" : " ");  printf("\nSorted Array:\n");  bogosort(x, 6);  for (i=0; i < 6; i++) printf("%d ", x[i]);  printf("\n");  } | Replace< with:  != |
| 15. | sorted\_list.c | for(i=1; i<size; ){  if(array[i-1] <= array[i])  ++i;  else{  tmp = array[i];  array[i] = array[i-1];  array[i-1] = tmp;  --i;  if(i == 0)  i = 1;  }} | Replace<= with:  < |
| 16. | sorted\_list.c | for(i=1; i<size; ){  if(array[i-1] <= array[i])  ++i;  else{  tmp = array[i];  array[i] = array[i-1];  array[i-1] = tmp;  --i;  if(i == 0)  i = 1;  }} | Replace ++i with:  i++ |
| 17. | bubble\_two.c | while (s) {  s = 0;  for (i = 1; i < j; i++) {  if (x[i] < x[i - 1]) {  t = x[i];  x[i] = x[i - 1];  x[i - 1] = t;  s = 1;  }  }  j--;  } | Repace 1 with any non-zero number:  s=2; |
| 18. | bubble\_two.c | while (s) {  s = 0;  for (i = 1; i < j; i++) {  if (x[i] < x[i - 1]) {  t = x[i];  x[i] = x[i - 1];  x[i - 1] = t;  s = 1;  }  }  j--;  } | Replace< with:  != |
| 19. | bubble\_two.c | while (s) {  s = 0;  for (i = 1; i < j; i++) {  if (x[i] < x[i - 1]) {  t = x[i];  x[i] = x[i - 1];  x[i - 1] = t;  s = 1;  }  }  j--;  } | Replace< with:  <= |
| 20. | bubble\_two.c | while (s) {  s = 0;  for (i = 1; i < j; i++) {  if (x[i] < x[i - 1]) {  t = x[i];  x[i] = x[i - 1];  x[i - 1] = t;  s = 1;  }  }  j--;} | Replace j-- with:  --j |
| 21. | sum\_one.c | int sum (int a, int b)  {  int s;  s=a+b;  return s;  } | Modify the code to:  return a+b; |
| 22. | countPositive.c | int countPositive (int x[], int length)  {  int count = 0;  int i;  for (i = 0; i < length; i++) {  if (x[i] >= 0) {  count++;  }  }  return count;  } | Replace< with:  != |
| 23. | 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< with:  != |
| 24. | oddOrPos.c | int oddOrPos (int x[], int length)  {  int count = 0;  int i;  for (i = 0; i < length; i++) {  if (x[i] % 2 == 1 || x[i] % 2 == -1 || x[i] > 0) {  count++;  }  }  return count;  } | Delete the first condition:  if ( x[i] % 2 == -1 || x[i] > 0) |
| 25. | oddOrPos.c | int oddOrPos (int x[], int length)  {  int count = 0;  int i;  for (i = 0; i < length; i++) {  if (x[i] % 2 == 1 || x[i] % 2 == -1 || x[i] > 0) {  count++;  }  }  return count;  } | Replace< with:  != |
| 26. | bubble.c | for (c = 0 ; c < ( n - 1 ); c++)  {  for (d = 0 ; d < n - c - 1; d++)  {  if (array[d] > array[d+1]) {  swap = array[d];  array[d] = array[d+1];  array[d+1] = swap;  }  }  } | Replace< with:  != |
| 27. | bubble.c | for (c = 0 ; c < ( n - 1 ); c++)  {  for (d = 0 ; d < n - c - 1; d++)  {  if (array[d] > array[d+1]) {  swap = array[d];  array[d] = array[d+1];  array[d+1] = swap;  }  }  } | Replace< with:  != |
| 28. | bubble.c | for (c = 0 ; c < ( n - 1 ); c++)  {  for (d = 0 ; d < n - c - 1; d++)  {  if (array[d] > array[d+1]) {  swap = array[d];  array[d] = array[d+1];  array[d+1] = swap;  }  }  } | Replace> with:  >= |
| 29. | max.c | int max(int x, int y)  {  if (x > y)  return x;  else  return y;} | Replace> with:  >= |
| 30. | findVal.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< with:  != |