SFT – WORKSHOP6

NAME: Chaerin Yoo

ID: 102998234

Email: [cyoo10@myseneca.ca](mailto:cyoo10@myseneca.ca)

Section: NCC

Stringhelp.c

|  |  |
| --- | --- |
| Original Code that has a bug | Fixed Code |
| int nextWhite(const char\* str)  {  int i, result = -1; for (i = 0; result < 0 && str[i] != '\0'; i++)   { if (str[i] == ' ' || str[i] == '\t')   { result = i;   }   } return ((result \* -1) != result) ? i : result; } | int nextWhite(const char\* str)  {  int i, result = -1;  for (i = 0; result < 0 && str[i] != '\0'; i++)  {  if (isspace(str[i]))  {  result = i;  }  }  return (result < 0) ? i : result;  } |

**What was wrong with the lines**: The original code had an issue where it would return an index that is one past the end of the string if no whitespace character was found. Specifically, the return statement (result < 0) ? i : result could cause i to be out of bounds.

**How I fixed it?** I used a code review technique to analyze the loop and the return statement logic, identifying that i would be incremented past the end of the string when no whitespace is found. This allowed me to see that the return value could be out of bounds.

|  |  |
| --- | --- |
| Original Code that has a bug | Fixed Code |
| int isNumber(const char\* str, const int len)  {  int result;  for (result = 1; result <= len && result; result++)  {  if (!isdigit(str[result - 1])) result = -1;  }  return result;  } | int isNumber(const char\* str, const int len)  {  int i, result = 1;  for (i = 0; i < len && result; i++)  {  if (!isdigit(str[i]))  {  result = 0;  }  }  return result;  } |

**What was wrong with the lines**: The original code causes the result to surpass the string's length by wrongly increasing it inside the loop, thereby regulating the loop and storing the result.

**How I fixed it?** Using code assertation, I saw that result was being misused as a result flag as well as a loop control variable.

|  |  |
| --- | --- |
| Original Code that has a bug | Fixed Code |
| struct StringIndex indexString(const char\* str)  {  struct StringIndex result = { {0}, {0}, {0}, 0, 0, 0  };  char buf[MAX\_WORD\_SIZE + 1] = { 0 };  int i, sp; | struct StringIndex indexString(const char\* str)  {  struct StringIndex result = { {0}, {0}, {0}, 0, 0, 0 };  int i, sp;  strcpy(result.str, str); |

**What was wrong with the lines**: The original code lacks required logic or initiation actions to properly populate the resultant structure. The str field in the output is not being replicated, which can cause erroneous or unclear behavior using result.str.

**How I fixed it?** Using code review and assertions, I looked at the loops' and conditions' logical flow.

|  |  |
| --- | --- |
| Original Code that has a bug | Fixed Code |
| struct StringIndex indexString(const char\* str)  {  struct StringIndex result = { {0}, {0}, {0}, 0, 0, 0 };  char buf[MAX\_WORD\_SIZE + 1] = { 0 };  int i, sp;  strcpy(result.str, str);  if (str[0] != '\0')  {  result.lineStarts[0] = 0;  result.numLines = 1;  }  for (i = 0; str[i] != '\0'; i++)  {  while (str[i] != '\0' && isspace(str[i]))  {  if (str[i] == '\0')  {  result.lineStarts[result.numLines++] = i + 1;  }  i++;  } | struct StringIndex indexString(const char\* str)  {  struct StringIndex result = { {0}, {0}, {0}, 0, 0, 0 };  int i, sp;  strcpy(result.str, str);  if (str[0] != '\0')  {  result.lineStarts[0] = 0;  result.numLines = 1;  }  for (i = 0; str[i] != '\0'; i++)  {  while (str[i] != '\0' && isspace(str[i]))  {  if (str[i] == '\n')  {  result.lineStarts[result.numLines++] = i + 1;  }  i++;  } |

**What was wrong with the lines**: The condition if (str[i] == '\0') inside the nested while loop is false since it checks for the null terminator within a loop already intended to end at the null terminator.

**How I fixed it?** I used assertions to examine the logical flow of the loops and conditions.

|  |  |
| --- | --- |
| Original Code that has a bug | Fixed Code |
| void getNumber(char word[], const struct StringIndex\* idx, int numberNum)  {  int i, sp, start;  word[0] = '\0';  if (numberNum < idx->numNumbers && numberNum >= 0)  {  start = idx->numberStarts[numberNum];  sp = nextWhite(idx->str + start);  for (i = 0; i < sp; i++)  {  word[i] = idx->str[sp + i];  }  word[sp] = '\0';  }  } | void getNumber(char word[], const struct StringIndex\* idx, int  numberNum)  {  int i, sp, start;  word[0] = '\0';  if (numberNum < idx->numNumbers && numberNum >= 0)  {  start = idx->numberStarts[numberNum];  sp = nextWhite(idx->str + start);  for (i = 0; i < sp; i++)  {  word[i] = idx->str[start + i];  }  word[sp] = '\0';  }  } |

**What was wrong with the lines**: In the original code, the line word[i] = idx->str[sp + i]; incorrectly uses sp + i as the index for idx->str, which leads to copying the wrong segment of the string.

**How I fixed it?** I used assertions to carefully examine the logic of the loop and the indexing of the string.