|  |  |  |  |
| --- | --- | --- | --- |
| Function | Expected | Result | Fix |
| b\_addc(); | No Run-Time error when init\_capacity is not between 0 and datatype size  No memory leak  Buffer mode is f,a,m  inc\_factor can only be a 0, not 0, for FIXED, and between two numbers for for ADD, and MULT modes,  returns pointer to Buffer with updated contents | Throws run-time error for overflow (negative because signed short)  Memory leak from malloc pBuffer->cb\_head  printf(“%c”, pBuffer->mode)  printf(“%d”, inc\_factor)  printf(“%d”, pBuffer->inc\_factor)  using printf to display contents, Buffer has been allocated & updated | 0 < init\_capacity < sizeof(short)  free(pBuffer->cb\_head)  free(Buffer)  none |
| b\_addc(); | FIXED MODE, nothing happens  ADD MODE, add Buffer inc\_factor to Buffer capacity  MULT MODE, add Buffer inc\_factor to Buffer capacity as a percent  valid memory allocation of cb\_head  change flags if memory allocation is different  old buffer has new allocated memory  added new symbol to new Buffer character array  addc\_offset increased by 1 per symbol  buffer has a new capacity | return NULL  inc\_factor between 0 and 32767, no OVERFLOW,  no OVERFLOW  checked through conditional statement  valid memory allocation of cb\_head  return NULL if memory allocation failed  pBD->flags = SET\_R\_FLAG;,  printf(“%d”, pBD->flags) to view flag  true  true, checked with b\_print  true, checked with b\_print  true, checked with printf() | none |
| b\_free(); | No memory leaks | No memory leaks | none |
| b\_isfull(); | Buffer is full | Returns 1 or 0 if full or if not full. | none |
| b\_limit(); | Buffer returns number of symbols in the Buffer character | When debugging printf(“%d”, pBD->addc\_offset) | none |
| b\_capacity(); | Buffer returns capacity | true | none |
| b\_mark(); | return the value of markc\_offset  return -1 on error | true, changing markc\_offset and displaying it through printf reveals a change in markc\_offset  true |  |
| b\_mode(); | return the mode of the Buffer | true | none |
| b\_incfactor(); | return inc\_factor  return 0x100 on error | true  true, call b\_incfactor when pBD does not have memory allocation | none |
| b\_load | Load file  Displays error for symbol one by one until the end of the file  Valid file  add symbol to character buffer  return number of symbols | true, get symbol through (char)fgetc(fi) , store symbol  true, printf("%c%d", symbol,symbol);  return LOAD\_FAIL  if (feof(fi)), breaks loop when end of file reached  addc(), and any errors will come from addc()  true, check with ass1.ai | none |
| b\_isempty(); | returns values based on empty or not | true with printf, and ass1e.ai | none |
| b\_getc(); | returns character in Buffer  check EOB | 99% true, increment getc\_offset for the next character to return.  1% false, returns all characters in a file (ass1.ai) except for the first character, and displays random characters at the end  true, flag is set to SET\_EOB when pBD->getc\_offset == pBD->addc\_offset | Buffer character is being returned one character Buffer ahead, return 1 index back in getc\_offset  \*(pBD->cb\_head+(pBD->getc\_offset-1))  none |
| b\_eob(); | return EOB flag | True, printf(“%c”, pBD->flag) to see where EOB is | none |
| b\_print(); | print symbols in character buffer  print empty list if empty  Return number of symbols printed  stop reading when end of buffer reached | True through printf using file ass1.ai  True through printf using file ass1e.ai  True through printf using file ass1.ai  true, BREAK; | None |
| b\_compact(); | Add symbol to a space after a compact location  Expand/compress capacity to a valid amount  reallocate memory for new capacity  memory allocation is valid  Buffer has an updated capacity | pBD->cb\_head[pBD->addc\_offset++] = symbol;  buffer\_new\_capacity = (pBD->addc\_offset + 1) \* sizeof(char);  buffer\_temp = (char \*)realloc(pBD->cb\_head, (unsigned short)(buffer\_new\_capacity));  memory allocation is different from current character buffer, Buffer is updated  true  true | check function by allocating a different addc\_offset to Buffer, overwriting existing symbols in the character Buffer when COMPACT. During expand use existing addc\_offset |
| b\_rflag(); | print current flag | ??? Displays the wrong flag when passing ass1.ai  actual Ass1.out displays hex FFFC but program only accepts hex FFFC  Ass1.ai displays 0 when actual Ass1.out says FFFC | Bitwise operation changed to pBD->flags = DEFAULT\_FALGS | SET\_EOB; for b\_getc() |
| b\_retract(); | returns / goes back 1 character backwards | True, before and after printf() of pBD->get\_offset | None |
| b\_reset(); | Returns updated character position to return its symbol | True, call function b\_mark and set mark, then call function b\_reset(), then call b\_print(). It will reprint a list at the location of mark | None |
| b\_getcoffset(); | Return the character offset of the Buffer | True, print in the function to view the current getc\_offset (in debug before final file version) before returning | none |
| b\_rewind(); | offset for character reading is reset | True, call b\_rewind(), then call b\_print(). The character buffer is displayed starting from \*cb\_head | none |
| b\_location(); | Return character at location indicated by loc\_offset | True, using print with chosen index call b\_location at any location.  Returns garbage character when index OVERFLOW | None |