Outline of SAIS algorithm:
Input:
1) vector<int> T: an original string S of characters is converted to an integer array (vector) T, such that the last integer in T is 0, and 0 is the smallest of all integers in T and does not occur anywhere else in T.
2) vector<int> SA: an integer array (vector) whose entries are initialized to -1; at the end of the execution, SA holds a suffix array for T (hence, for S).
3) int alphabetSize: is the size of alphabet in T that equals to the largest integer in T plus 1, the total number of distinct characters in T (including 0).
Output:
SA is passed by reference, so at the end of execution of this function SAIS, SA holds a suffix array for T (consequently for the original string S).
This is a recursive function: it must have a termination step and a recursive call inside it.

Note: the code inside boxes is the output in my detailed-output files that are created for you to debug your program.

SAIS(vector<int> &T, vector<int> &SA, int alphabetSize){

```
cout << "Iteration " << iter << endl;
cout << "T at iteration " << iter << endl;
for(int i = 0; i < sSize; i++)
       cout << T[i] << " " ;
cout << endl;
```

0) **Termination condition:** Check if size of T is equal to alphabetSize, and if so do this:
- Scan T from Left-to-Right using index i = 0, 1, 2…, and set:
  SA [T [i]] = i;
- Return.

1) Do Steps 0-1 of this document, including:
- Calculate arrays A, C and B for T;
- Calculate array *t*, holding types (L or S) of suffixes of T;
- Do Steps 0 and 1.

```
cout << "LMS prefixes are sorted.\nSA at iteration " << iter << endl;
       for(int i = 0; i < sSize; i++)
              cout << SA[i] << " " ;
       cout << endl;
```

2) Do Step 2:

- Give integer-names to LMS substrings and calculate array N (let the largest integer name be ***largest***);
- Build shortened string T1 using names in N;

```
cout << "T1 at iteration " << iter << endl;
for(int i = 0; i < newAlphSize; i++)
       cout << T1[i] << " " ;
cout << endl;
```

- Declare and initialize SA1 (same size as T1, and initialized to -1).

3) Make a recursive call on T1 and SA1 and alphabet size of T1:

```
cout << "Iteration " << iter << ". Recursive call on T1."
          <<" \n*****************************************" << endl;
```

SAIS(T1, SA1, largest + 1); //recursive call

```
cout << "\n*****************************************"
       << "\nAfter recursive call inside iteration " << iter << endl;
cout << "SA1 at iteration " << iter << " (after recursive call) is:" << endl;
for(int i = 0; i < newAlphSize; i++)
       cout << SA1[i] << " " ;
cout << endl;
```

4) **After the recursive call:** Do **Step 4** of this document:

- Initialize SA with -1; //again
- Overwrite T1 with positions of LMS substrings in T from Left-to-Right
- Place positions of LMS substrings into SA using their relative order from SA1;
- Repeat Step 1 of this document.

```
cout << "SA at the end of SAIS function just before the return at iteration "
       << iter <<endl;
for(int i = 0; i < sSize; i++)
       cout << SA[i] << " " ;
cout << endl;
return ;
```

}//end of SAIS