Task 1 Algorithm

* Include the headers
  + Iostram
  + Fstream
  + Iomanip
  + Cstring
    - For the use of strlen later in the program
* Function prototype
  + Char examGrade (int score)
* In Main ()
  + Declare the variables
    - Int I, score, len
    - Int numberOfQuestions = 20
      * Total number of questions
    - Char ch
    - Char ID [9]
      * Array to hold ID
    - Char \*keystring
      * Char pointer for answer
    - Char \*answerString
      * Char student answer
  + Open file stream
  + Open input file
  + If the input file cannot be opened print an error screen to the user and return 1
  + Set up dynamic arrays
    - Keystring = new char[noOfQuestions + 1]
    - answerString = new char[noOfQuestions + 1]
  + print to the screen
    - processing data
  + read the answer key from the input file
    - inFile.get(keyString, noOfQuestions + 1)
  + output the answers
    - cout << “Key: “ << keystring << endl << endl
  + formatting
    - cout << left << setw(15) << "Student ID" << setw(25) << "Student Answers" << setw(10) << "Score" << setw(10) << "Grade" << endl;
  + pull from the input file
    - inFile >> ID;
    - inFile.get(ch)
  + inFile.get(answerString, noOfQuestions + 1)
  + while loop
    - len = strlen(answerString);
    - for (i = len; i < noOfQuestions; i++)
      * answerString[i] = ' ';
    - answerString[noOfQuestions] = '\0';
    - score = 0;
    - for (i = 0; i < noOfQuestions; i++)
    - if (answerString[i] != ' ')
      * if (answerString[i] == keyString[i])
        + score = score + 2;

if the answer is correct

* + - * Else
        + score = score - 1;

if the answer is wrong

* + - cout << left << setw(15) << ID << setw(25)<< answerString << setw(10) << score << setw(10) << examGrade(score) << endl;
      * formatting
    - inFile >> ID;
    - inFile.get(ch);
    - inFile.get(answerString, noOfQuestions + 1);
      * pull the same information as before the while loop
  + close input file
  + delete arrays
    - delete [] Keystring;
    - delete [] answerString
* in Char examGrad (int score)
  + switch statement
    - case 0:
    - case 1:
    - case 2:
    - case 3:
    - case 4:
    - case 5:
      * return 'F';
    - case 6:
      * return 'D';
    - case 7:
      * return 'C';
    - case 8:
      * return 'B';
    - case 9:
    - case 10:
      * return 'A';
    - default:
      * return 'F';
  + NOTE: This switch statement returns the total grade for the student when called in the final cout of the while loop in the main function

Task 2

* Include headers
  + Iostream
  + Iomanip
  + String
* Function prototypes
  + int sumVotes(int list[], int size);
  + int winnerIndex(int list[], int size);
* in int main()
  + declare variables
  + declare pointers
  + ask the user for number of candidates
  + take the user’s input
  + declare dynamic arrays
    - candidates = new string[numberOfCandidates];
    - votes = new int[numberOfCandidates];
  + use the following formatting
    - cout << fixed << showpoint;
    - cout << setprecision(2);
  + ask for the name and number of votes
  + for loop
    - use a for loop to store the user’s input in the dynamic arrays
      * for (i = 0; i < numberOfCandidates; i++)
      * cin >> candidates[i] >> votes[i];
  + update the total number of votes
    - totalVotes = sumVotes(votes, numberOfCandidates);
  + table formatting
    - cout << "Candidate votes Recieved % of Total Votes" << endl;
  + use a similar loop that took the user’s input to print the results
    - for (i = 0; i < numberOfCandidates; i++)
      * cout << left << setw(10) << candidates[i] << right << " " << setw(10) << votes[i] << " " << setw(15) << (static\_cast<double> (votes[i])/ static\_cast<double>(totalVotes)) \* 100 << endl;
  + Print out the total number of votes
  + Print out the winner
  + Delete the arrays
    - delete [] candidates;
    - delete [] votes;
* in int sumVotes
  + int sumVotes(int list[], int size)
    - to calculate the sum
  + declare the local variable
  + add up all the components when called.
    - for (int i =0; i < size; i++)
    - sum = sum + list[i];
  + return the new calculated sum to main
* in winnnerIndex
  + int winnerIndex(int list [], int size)
    - to calculate/find the winner
  + declare the local variables
  + use a for loop similar to the sum for loop to find the winner
    - for (int i = 0; i < size; i++)
    - if (list[i] > list[winInd])
    - winInd = i;
  + return the winner index.