|  |  |  |
| --- | --- | --- |
| **Name:**  **P K Navin Shrinivas** | **SRN:PES2UG20CS237** | **Section:g(B-2)** |
| **Date:03/06/21** | **Week Number:5** |

|  |  |
| --- | --- |
| **1** | 1)Write functions to  a) Reverse a string.  b) Check for equality of strings.  **Input1:**  Enter string  abbcbba  **Output1:**  Reversed string is = abbcbba  Given string is abbcbba is palindrome  **Input2:**  Enter string  hi  **Output2:**  Reversed string is = ih  Given string is hi is not palindrome |
|  | **Program:**  **1\_1.c:**  #include<stdio.h>  void reversestring(char\* *rev*,char\* *c*)  {  int n=0;  while(\*(c+n)!='\0'){  n++;  }  for(int i=0;i<n;i++)  \*(rev+n-i-1)=\*(c+i);  \*(rev+n)='\0';  }  bool palindrome(char\* *c*)  {  int n=0;  while(\*(c+n)!='\0'){  n++;  }  char rev[n];  reversestring(rev,c);  for(int i=0;i<n;i++)  {  if(rev[i]!=c[i])  return false;  }  return true;  }  **1\_1.h:**  void reversestring(char\* *rev*,char\* *c*);  bool palindrome(char\* *c*);  **1prog.c:**  #include<stdio.h>  #include<stdbool.h>  #include"1\_1.h"  int main(){  char c[100000];  char rev[100000];  printf("Enter string : ");  scanf("%[^\n]%c",c);  reversestring(rev,c);  /\*need to pass size of array here are in fucntion its considered as a pointer  we also pass in the empty reverse array as returning local variables from function  can lead to segfaults...This is due to scope of variabe\*/  printf("Reverse of string : ");  int i=0;  while(rev[i]!='\0')  {  printf("%c",rev[i]);  i++;  }  printf("\n");  if(palindrome(c))  printf("The given string is a palindrome");  else  printf("Nope!Not a palidorme!");  } |
|  | **Output Screenshot:**  **1** |
| **2** | Write function to find all occurrences of a character in a string and use this function to replace all occurences of a character by specific character.  Input1:  Enter the string : Welcome to C programming  Enter a character to replace: o  Enter character to replace with r : @  Output1:  Before replace: Welcome to C programming  After replace: Welc@me t@ C pr@gramming |
|  | **Program:**  **2\_1.c:**  void charreplace(char\* *c*,char *r*,char *w*)  {  int n=0;  while(\*(c+n)!=int lookup[123]={0};  /\*this pesudo implementation of hash tables will allow us  to solve this problem in O(n) instead of O(n^2)worst case senarios\*/  /\*think of this as simply a map\*/  *//n is the number of charecters in the sentence'\0'){*  n++;  }  for(int i=0;i<n;i++)  {  if(\*(c+i)==r)  \*(c+i)=w;  }  }  **2\_1.h:**  void charreplace(char\* *c*,char *r*,char *w*);  **2prog.c:**  #include<stdio.h>  #include<stdbool.h>  #include"2\_1.h"  int main()  {  char c[100000],r,w;  printf("Enter a sentence :");  scanf("%[^\n]%\*c",c);  printf("Enter letter to replace :");  scanf("%c",&r);  printf("Enter letter to replace %c with :",r);  scanf("\t%c",&w);  charreplace(c,r,w);  printf("After replacing :");  int i=0;  while(c[i]!='\0')  {  printf("%c",c[i]);  i++;  }  } |
|  | **Output Screenshot:**  **2** |
| **3** | Write a function to remove all repeated characters from a given string and display the string without duplicate characters.  **Input 1:**  Enter any string: hello world  **Output 1:**  String before removing duplicates: hello world  String after removing duplicates: helo wrd  **Input 1:**  Enter any string: programming in c  **Output 1:**  String before removing duplicates: programming in c  String after removing duplicates: progamin c |
|  | **Program:**  **3\_1.c:**  void noduplicate(char\* *out* , char\* *c*)  {  int lookup[123]={0};  /\*this pesudo implementation of hash tables will allow us  to solve this problem in O(n) instead of O(n^2)worst case senarios\*/  /\*think of this as simply a map\*/  *//n is the number of charecters in the sentence*  int i=0;  while(\*(c+i)!='\0')  {  lookup[(int)\*(c+i)]++; *//hashing*  i++;  }  i=0;  int w=0;  while(\*(c+i)!='\0')  {  if(lookup[(int)\*(c+i)]==0)  i++;  else if(lookup[(int)\*(c+i)]>1)  {  \*(out+w)=\*(c+i);  lookup[(int)\*(c+i)]=0;  i++;  w++;  }  else{  \*(out+w)=\*(c+i);  lookup[(int)\*(c+i)]=0;  i++;  w++;  }  }  }  **3\_1.h:**  void noduplicate(char\* *c* , char\* *out*);  **3prog.c:**  *//O(n) solution to a duplicate charecter problem*  /\*this pesudo implementation of hash tables will allow us  to solve this problem in O(n) instead of O(n^2)worst case senarios\*/  /\*think of this as simply a map\*/  *//n is the number of charecters in the sentence*  #include<stdio.h>  #include"3\_1.h"  int main(){  char c[10000];  printf("Enter a string :");  scanf("%[^\n]%\*c",c);  char ans[10000];  noduplicate(ans,c);  int i=0;  while(ans[i]!='\0')  {  printf("%c",ans[i]);  i++;  }  } |
|  | **Output Screenshot:**  **3** |
| 4 | Write function to Concatenate two strings and use this to concatenate n (i.e, say 2) strings.  **Input 1:**  Enter 1st string  pes  Enter 2nd string  university  Enter number of times u want to append  1  **Output1:**  Concatenated string is pesuniversity  **Input2:**  Enter 1st string  pes  Enter 2nd string  university  Enter number of times u want to append  2  **Output2:**  Concatenated string is pesuniversityuniversity |
|  | **Program:**  **4\_1.c:**  void concatenate(char\* *s1*,char\* *s2*,int *n*)  {  int size1=0;  while(\*(s1+size1)!='\0'){  size1++;  }  int size2=0;  while(\*(s2+size2)!='\0'){  size2++;  }  while(n)  {  for(int i=0;i<size2;i++)  {  if(\*(s2+i)!='\0')  {  \*(s1+size1)=\*(s2+i);  size1++;  }  }  n--;  }  \*(s1+size1)='\0';  }  **4\_1.h:**  void concatenate(char\* *s1*,char\* *s2*,int *n*);  **4prog.c:**  #include<stdio.h>  #include"4\_1.h"  int main(){  char s1[10000],s2[10000];  int n;  printf("Enter string 1 :");  scanf("%[^\n]%\*c",s1);  printf("Enter a string 2 :");  scanf("%[^\n]%\*c",s2);  printf("Enter number of times to concatenate :");  scanf("%i",&n);  concatenate(s1,s2,n);  int i=0;  while(s1[i]!='\0')  {  printf("%c",s1[i]);  i++;  }  } |
|  | Output Screenshot:  4 |
| 1 | **Practice Programs**  Write a function to count the number of occurrences of a given character. Use this to find the number of occurrences of every character in a word.  **Input:**  pesit pes!  **Output:**  i occurs is 1 times  t occurs is 1 times  occurs is 1 times  p occurs is 2 times  e occurs is 2 times  s occurs is 2 times  ! occurs is 1 times |
|  | **Program:**  **5\_1.c:**  #include<stdio.h>  void ocurrence(char\* *c*)  {  int lookup[123]={0};  /\*this pesudo implementation of hash tables will allow us  to solve this problem in O(n) instead of O(n^2)worst case senarios\*/  /\*think of this as simply a map\*/  *//n is the number of charecters in the sentence*  int i=0;  while(\*(c+i)!='\0')  {  lookup[(int)\*(c+i)]++; *//hashing*  i++;  }  i=0;  while(\*(c+i)!='\0')  {  if(lookup[(int)\*(c+i)]==0)  i++;  else if(lookup[(int)\*(c+i)]>1)  {  printf("%c occurs : %i times \n",\*(c+i),lookup[(int)\*(c+i)]);  lookup[(int)\*(c+i)]=0;  i++;  }  else{  printf("%c occurs : %i time \n",\*(c+i),lookup[(int)\*(c+i)]);  lookup[(int)\*(c+i)]=0;  i++;  }  }  }  **5\_1.h:**  void ocurrence(char\* *c*);  **5prog.c:**  /\*this pesudo implementation of hash tables will allow us  to solve this problem in O(n) instead of O(n^2)worst case senarios\*/  /\*think of this as simply a map\*/  *//n is the number of charecters in the sentence | similar to program 3 of week 5*  #include<stdio.h>  #include"5\_1.h"  int main(){  char c[10000];  printf("Enter a string :");  scanf("%[^\n]%\*c",c);  char ans[10000];  ocurrence(c);  } |
|  | **Output Screenshot:**  5 |
| 2 | Write the function strend (s , t ), which returns 1 if the string t occurs at the end of the string s, and zero otherwise.  **Input1:**  hello world!  world  **Output 1:**  0  **Input2:**  hello world! world  world  **Output 2:**  1 |
|  | **Program:**  **6\_1.c:**  #include<stdbool.h>  #include<stdio.h>  bool postfix(char\* *c*,char \**b*)  {  int sc=0;  while(\*(c+sc)!='\0'){  sc++;  }  int sb=0;  while(\*(b+sb)!='\0'){  sb++;  }  sb--;  for(int i=sc-1;i>=0;i--)  {  if(\*(b+sb)==\*(c+i))  sb--;  else if (sb>0 && \*(b+sb)!=\*(c+i))  return false;  }  return true;  }  **6\_1.h:**  bool postfix(char\* *c*,char \**b*);  **6prog.c:**  #include<stdio.h>  #include<stdbool.h>  #include"6\_1.h"  int main(){  char c[10000];  printf("Enter string 1 : ");  scanf("%[^\n]%c",c);  char b[10000];  printf("Enter string 2 : ");  scanf("\t%[^\n]%c",b);  if(postfix(c,b))  printf("1");  else  printf("0");  } |
|  | **Output Screenshot:**  6 |