Date

Problem Statement : 1

Write a program to insert a substring into a string from a particular position.

Assumptions:

We have stored string in character array (string) a and substring in character array (string) b. We are storing length of a and b in integer l1 and l2 respectively, position in integer p.

Algorithm:

Step 1: Start
Step 2: Input a and b.
Step 3: Input p.
Step 4: Assign length of a to l1 and length of b to l2, 0 to c.
Step 5: Assign l1 to i, check if i is greater than equal to p-1, repeat step 6 and decrease i by 1. Otherwise go to step 7.
Step 6: Assign a[i] to a[i+l2]
Step 7: Assign p-1 to i, check if i is less than p-1+l2, repeat step 8 and increase i by 1 after each repetition.
Step 8: Assign b[c] to a[i] and increase c by 1.
Step 9: Print a (new string).
Step 10: Stop

Swati

Date

Problem Statement : 2

Write a C code that loops over the string and replace each character to the character immediately preceeding it in the alphabet.

Example : Input string : HELLO

Output String : GDKKN.

Assumptions :

We have stored string in character array (string) a

Algorithm :

Step 1 : Start

Step 2 : Input a.

Step 3 : Assign 0 to i, check if a[i] is not equal to '\0', repeat step 4 to step 8, increment i by 1.

Step 4 : Check if a[i] is equal to 'a', go to step 5 otherwise goto step 6

Step 5 : Assign 'z' to a[i]. Goto step 3

Step 6 : check if a[i] is equal to 'A', go to step 7 otherwise goto step 8

Step 7 : Assign 'Z' to a[i]. Goto step 3.

Step 8 : Assign a[i]-1 to a[i].

Step 9 : Print a (output string).

Step 10 : Stop.

Swati

Date

Problem Statement : 3

Write a program that takes your full name (First, middle and last) as input and displays the abbreviations of the first and middle names except the last name which is displayed as it is.

For e.g. : Input : Aman Singh Rawat

Output : A. S. Rawat

Assumptions :

We have stored name in character array (string) a, size of name in integer l.

Algorithm :

Step 1 : Start

Step 2 : Input a.

Step 3 : Assign length of a to l.

Step 4 : Assign 0 to i, check if i is less than l, repeat step 5 to step 13. Increment i by 1 after each repetition.

Step 5 : Assign i+1 to j, check if j is less than equal to l, repeat step6 to step12. Increment j after each repetition.

Step 6 : If a[j] is equal to 32, go to step 7 otherwise goto step 9.

Step 7 : Print a[i].

Step 8 : Assign j+1 to i.

Step 9 : If a[j] is equal to '\0' go to step 10 otherwise goto step 5

Step 10 : Assign i to k, check if k is less than l, repeat step 11. Increment k after each repetition.

Step 11 : Print a[k].

Step 12 : Go to step 13

Step 13 : if a[j] is equal to '\0' go to step 14.

Step 14 : Stop.

Swati

Date

Problem Statement A :

Write a program to check if the 2 strings entered by user are anagrams or not. 2 words are said to be anagrams if the letters of one word can be rearranged to form the other word.
e.g. HEART and EARTH are anagrams of each other.

Assumptions :

We have stored the first string in character array (string) a and second string in character array (string) b, length of ~~f~~ a in int l1 and length of b in integer l2.

Algorithm :

Step 1 : Start
Step 2 : Input a and b.
Step 3 : Assign length of a to l1 and length of b to l2.
Step 4 : Assign 0 to c.
Step 5 : If l1 is equal to l2 goto step 6 otherwise goto step 14
Step 6 : Assign 0 to i, check if i is less than l1, repeat step 7 to step 12, increment i by 1 after each repetition.
Step 7 : Assign 0 to j, check if j is less than l2, repeat step 8 to step 12, increment j by 1 after each repetition.
Step 8 : If a[i] is equal to b[j] goto step 9 otherwise goto step 7.
Step 9 : Increment c by 1.
Step 10 : Assign j to l, check if l is less than l2, repeat step 11, increment l by 1 after each repetition.
Step 11 : Assign b[l+1] to b[l].
Step 12 : Decrement j by 1, decrement l2 by 1.

Swati

Date _____

Step 13 : If c is equal to 11 print "Strings are anagrams"
otherwise print "Strings are not anagrams". Goto step 15

Step 14 : Print "Strings are not anagrams".

Step 15 : Stop.

Date

Problem Statement : 5

Write a program using dynamic memory allocation to insert elements in an array and perform the following operation:

i) Searching of an element.

ii) Replacing the searched element with its cube and print the array.

NOTE: Make 2 different user defined function to perform the operation.

Assumptions:

We have made 2 functions, search and cube. Size of array & element to be searched is stored in integer n and t respectively. Address of dynamically allocated array is stored in integer pointer p.

Algorithm:

Step 1: Start [Main Function]

Step 2: Input n.

Step 3: Dynamically allocate integer array of size n using malloc. p stores the address of the first element of the array.

Step 4: Assign 0 to i, check if i is less than n, repeat step 5, increment i by 1 after each repetition.

Step 5: Input (p+i).

Step 6: Input t.

Step 7: Call function search (p, n, t).

Step 8: Call function cube(p, n, t).

Step 9: Assign 0 to i, check if i is less than n, repeat step 10, increment i by 1 after each repetition.

Step 10: Print *(p+i).

Step 11: Stop

Swati

Date

[Search Function]

Step 1: Start

Step 2: Assign 0 to i, check if i is less than n, repeat step 3 to step 4. Increment i by 1 after each repetition.

Step 3: Check if *(p+i) is equal to a goto step 4 otherwise goto step 2.

Step 4: Print i

Step 5: Stop

[cube function]

Step 1: Start

Step 2: Assign 0 to i, check if i is less than n, repeat step 3 to step 4. Increment i by 1 after each repetition.

Step 3: Check if *(p+i) is equal to a go-to step 4 otherwise goto step 2.

Step 4: Assign (*(p+i) * *(p+i) * *(p+i)) to (*(p+i)).

Step 5: Stop.

Swati

Date

Problem Statement : 6

Write a program to reverse the digits of a number using pointer.

Assumptions :

We have stored number in integer n, reverse of number in integer r, address of n in integer pointer p.

Algorithm :

Step 1 : Start

Step 2 : Input n.

Step 3 : Assign address of n to p.

Step 4 : Check if (*p) is greater than 0, repeat step 5 to step 6.

step 5 : Assign (r * 10 + (*p) % 10) to r.

Step 6 : Assign (*p / 10) to *p.

Step 7 : Print "Reverse of number is".

Step 8 : Print r.

Step 9 : Stop.

Swati

Date

Problem Statement : 7

Write a program to add elements of 2 unequal size array into 3rd array using dynamic memory allocation.

Assumptions:

We have stored size of first and second array in integer n and m. Integer pointers a, b and c store the address of the first element of first array, second array and third array respectively.

Algorithm:

Step 1 : Start

Step 2 : Input n and m.

Step 3 : Check if n is greater than m goto step 4 otherwise goto Step 5

Step 4 : Dynamically allocate an ~~array of size~~ integer array of size n using calloc() and the address of the first element is stored in a. Dynamically allocate integer array of size n using calloc() and the address of the first element is stored in b. Goto step 6.

Step 5 : Dynamically allocate 2 integer array of size m using calloc() and the address of the first element of first array is stored in a, and address of 1st element of second array is stored in b.

Step 6 : Assign 0 to i, check if i is less than n, repeat step 7 Increment i after each repetition.

Step 7 : Input (a+i).

Step 8 : Assign 0 to i, check if i is less than m, repeat step 9, increment i after each repetition.

Step 9 : Input (b+i).

Swati

Date

Step 10 : Dynamically allocate integer array of size (n+m) using calloc() and the address of the first element is stored in c.

Step 11 : Assign 0 to i, check if i is less than n or i is less than m, repeat step 12. Increment i by 1 after each repetition.

Step 12 : Assign (*(a+i) + *(b+i)) to (*(c+i)).

Step 13 : Assign 0 to i, check if i is less than n or i is less than m, repeat step 14. Increment i by 1 after each repetition.

Step 14 : stop.

Swati

Date

Problem Statement : 8

Define a structure to store the roll no, name, age (between 11 to 14) and address of students. Input and store records of more than 10 students.

Write a function to print the names of all the students having age 14 and even roll number.

Assumptions :

Age and roll number are stored in integer age and r_n. Name and address is stored in character array name and address. All these variables are stored in a structure called student.

Number of students is stored in integer n. s is ~~stud.~~ array of structure student.

Algorithm :

Step 1 : Start [Main function]

Step 2 : Input n.

Step 3 : Assign 0 to i, check if i is less than n, repeat step 4 to ~~step 4~~ : step 5. Increment i by 1 after each repetition.

Step 4 : Input s[i].age, s[i].r_n.

Step 5 : Input s[i].name, s[i].address

Step 6 : Call function display. [display (s,n)].

Step 7 : Stop.

Step 1 : Start [display function]

Step 2 : Assign 0 to i, check if ~~is~~ i is less than s, repeat step 3-4 Increment i by 1 after each repetition.

Step 3 : Check if a[i].age is equal to 14 and (a[i].r_n)% 2 is equal to 0 goto step 4 otherwise goto step 2.

Swati

Date

Step 4: Print a[i].name . Print "\n".

Step 5: Stop.

Swati

Date

Problem Statement : 9

Write a structure to store the names, salary and working hours per day (fixed at the time of joining) of 10 employees. Calculate the increased monthly salary depending on the working hours per day as given below: Print the names of all the employees along with their new salaries.

Hours of work per day :	8	10	>= 12
Increase in salary :	Rs. 2500	Rs 5000	Rs. 7500

Assumptions :

Name is stored in character array name. Salary & working hour is ~~store~~ stored in integer salary and w_h. All these variables are stored in a structure called employee. e is an array of type structure employee.

Algorithm :

Step 1 : Start

Step 2 : Assign 0 to i, check if i is less than 10, repeat step 3 and 4. Increment i by 1 after each repetition.

Step 3 : Input e[i].salary, e[i].w_h.

Step 4 : Input e[i].name.

Step 5 : Assign 0 to i, check if i is less than 10, repeat step 6 to step 10. Increment i by 1 after each repetition.

Step 6 : check if e[i].w_h is equal to 8, increment e[i].salary by 2500 otherwise goto step 7.

Step 7 : Check if e[i].w_h is equal to 10, increment e[i].salary by 5000 otherwise goto step 8.

Step 8 : Check if e[i].w_h is ~~equal~~ greater than equal to 12, increment e[i].salary by 7500 otherwise goto step 9

Swati

Date

Step 9 : Print " Salary of "

Step 10 : Print e[i].name. Print e[i].salary.

Step 11 : Stop.

Date

Problem Statement : 10

Write a program to read 20 integers in a file. Separate them into 2 different files prime.txt and nonprime.txt such that all prime numbers are copied in prime.txt and remaining numbers in nonprime.txt.

Assumptions :

We have taken FILE pointers fp, fe, fo. We are storing no. temporarily in integer t.

Algorithm :

Step 1: Start

Step 2: Create file "input.txt" with fopen in "wt" mode & assign it to fp.

Step 3: Create file "prime.txt" with fopen in "wt" mode & assign it to fe.

Step 4: Create file "nonprime.txt" with fopen in "wt" mode & assign it to fo.

Step 5: Assign 1 to i, check if i is less than equal to 2, repeat step 6 & 7. Increment i by 1 after each repetition.

Step 6: Input t.

Step 7: Print t in file through fp using putw(t,fp).

Step 8: Rewind fp.

Step 9: Assign 1 to i, check if i is less than equal to 20, repeat step , Increment i by 1 after each repetition.

Step 10: Assign getw(fp) to t.

Step 11: Assign 2 to j, check if j is less than equal to t by 2, repeat step 12 to step 13. Increment j by 1 after each repetition.

Step 12: Assign 0 to c.

Step 13: Check if t is divisible by j, increment c by 1 goto step 14.

Step 14: Check if c is equal to 0, Print t in fe by putw(t,fe) otherwise print t in fo by putw(t, fo).

Swati

Date

Step 15 : Close fp by fclose (fp).

Step 16 : Rewind fo and rewind fe.

Step 17 : Check if t with value getw (fe) is not equal to end of file.
Repeat Step 18.

Step 18 : Print t.

Step 19 : Check if t with value getw (fo) is not equal to end of file.
Repeat step 20.

Step 20 : Print t.

Step 21 : Close fo and fe by fclose (fo) and fclose (fe).

Step 22 : Stop.

Swati

Date

Problem Statement : 11

Write a program to create a file with some contents. Display its contents. Also print all the positions of a given alphabet (user input) in the file.

Assumptions :

We are storing content in character array (string) a & temporarily in character array (string) b. fp is pointer to FILE. c stores the character to be found, it is character variable. ~~l is~~ integer l stores length of string.

Algorithm :

Step 1 : Start

Step 2 : Create file "input.txt" by fopen( ) using "wt" mode. Assign it to fp.

Step 3 : Input a

Step 4 : Input c.

Step 5 : Print a to file using fputs(a, fp).

Step 6 : Assign length of string to l.

Step 7 : ~~Rewind~~ Rewind fp. ~~Print b.~~

Step 8 : Input b through file by fgets(b, l+1, fp).

Step 9 : Print b.

Step 10 : Check if t with value getc (fp) is not equal to end of file. Repeat step 11 and 12.

Step 11 : Check if c is equal to t, print i.

Step 12 Increment i by 1.

Step 13 : Close fp by fclose (fp).

Step 14 : Stop.

Swati