1. Word Analytics   
  
Input :   
  
Read a text file (no special formatting, just a plain ASCII text file) and print off the following details:   
  
1. Number of words   
2. Number of letters   
3. Number of symbols (any non-letter and non-digit character, excluding white spaces)   
4. Top three most common words (you may count "small words", such as "it" or "the")   
5. Top three most common letters   
6. Most common first word of a paragraph (paragraph being defined as a block of text with an empty line above it) (Optional bonus)   
7. Number of words only used once (Optional bonus)   
8. All letters not used in the document (Optional bonus)   
  
Please note that words are not case sensitive, meaning the word "Hello" is the same as "hello" and "HELLO".

|  |
| --- |
| #include<stdio.h> |
|  | #include<string.h> |
|  | #include<stdlib.h> |
|  | struct words |
|  | { |
|  | char str[100]; |
|  | int count; |
|  | struct words \* next; |
|  |  |
|  | }\*p,\*q,\*head,\*t; |
|  | void sort() // function to sort words accoding to their frequency |
|  | { |
|  | int temp; |
|  | char temps[60]; |
|  | p=head; |
|  | q=head->next; |
|  | while(p->next!=NULL) |
|  | { |
|  | q=p->next; |
|  | while(q!=NULL) |
|  | { |
|  | if(p->count<q->count) |
|  | { |
|  |  |
|  | temp=p->count; |
|  | p->count=q->count; |
|  | q->count=temp; |
|  | strcpy(temps,p->str); |
|  | strcpy(p->str,q->str); |
|  | strcpy(q->str,temps); |
|  | } |
|  | q=q->next; |
|  | } |
|  | p=p->next; |
|  | } |
|  | } |
|  | void insert(char \*a) //used for inserting unique words in to the word structure otherwise count is incremented |
|  | { |
|  | int flag=0; |
|  | struct words \* temp = (struct words \*)malloc(sizeof(struct words)); |
|  | if(head==NULL) |
|  | { |
|  | strcpy(temp->str,a); |
|  | temp->next=NULL; |
|  | temp->count=1; |
|  | head = temp; |
|  | q=head; |
|  | } |
|  | else |
|  | { |
|  | p=head; |
|  | while(p!=NULL) |
|  | { |
|  | if(strcmp(p->str,a)==0) |
|  | { |
|  | flag=1; |
|  | p->count+=1; //to increment count |
|  | break; |
|  | } |
|  | p=p->next; |
|  | } |
|  | if(flag==0) // if word is not present |
|  | { |
|  | strcpy(temp->str,a); |
|  | temp->count=1; |
|  | temp->next=NULL; |
|  | q->next=temp; |
|  | q=q->next; |
|  | } |
|  | } |
|  | } |
|  | int wordify(char \*str) // used separate words eg hello,there will be separated as hello and there it writtens the word count |
|  | { |
|  | int i,j=0,strt=0,end,k=0,wc=0; |
|  | for(i=0;str[i]!='\0';i++) |
|  | { |
|  | if (str[i]>=32&&str[i]<=47) |
|  | { |
|  | char \* b = (char \*)malloc(100); |
|  | end = i-1; |
|  | for(j=strt;j<=end;j++) |
|  | { |
|  | b[k]=str[j]; |
|  | k++; |
|  | } |
|  | b[k]='\0'; |
|  | strt = i+1; |
|  | wc++; |
|  | //printf("%s\n",b); |
|  | insert(b); |
|  | free(b); |
|  | k=0; |
|  | } |
|  |  |
|  | } |
|  | return wc; |
|  | } |
|  | int count=0; // for counting number of letters |
|  | char \* To\_Lower(char \*a,int alpha[]) // converts paragraph to lowecase and count the number of letters |
|  | { |
|  | int i; |
|  | for(i=0;a[i]!='\0';i++) |
|  | { |
|  | if(a[i]>=65&&a[i]<=90) |
|  | { |
|  | a[i]+=32; |
|  | alpha[a[i]-97]+=1; |
|  | count++; |
|  |  |
|  | } |
|  | else if(a[i]>=97&&a[i]<=122) |
|  | { |
|  | alpha[a[i]-97]+=1; |
|  | count++; |
|  | } |
|  | } |
|  | return a; |
|  | } |
|  | int symb\_count(char \*a) // count number of symbols |
|  | { |
|  | int sc=0,i; |
|  | for(i=0;a[i]!='\0';i++) |
|  | { |
|  | if(a[i]>=33&&a[i]<=47||a[i]==64) |
|  | { |
|  | sc++; |
|  | } |
|  | } |
|  | return sc; |
|  | } |
|  | void frequent\_letters(int lf[],int alpha[]) //sorting letters according to frequency |
|  | { |
|  | int i,j,temp; |
|  | for(i=0;i<26;i++) |
|  | { |
|  | for(j=i+1;j<26;j++) |
|  | { |
|  | if(lf[i]<lf[j]) |
|  | { |
|  | temp=lf[i]; |
|  | alpha[j]=i+97; |
|  | alpha[i]=j+97; |
|  | lf[i]=lf[j]; |
|  | lf[j]=temp; |
|  | } |
|  | } |
|  | } |
|  | } |
|  | void letters\_not\_used(int lf[]) |
|  | { |
|  | int i; |
|  | printf("4.Letters not used : "); |
|  | for(i=0;i<26;i++) |
|  | { |
|  | if(lf[i]==0) |
|  | { |
|  | printf("%c ",i+97); |
|  | } |
|  | } |
|  | printf("\n"); |
|  | } |
|  | int occured\_once() // returns no of words occured only once |
|  | { |
|  | int c=0; |
|  | p=head; |
|  | while(p->next!=NULL) |
|  | { |
|  | if(p->count==1) |
|  | { |
|  | c++; |
|  | } |
|  | p=p->next; |
|  | } |
|  | return c; |
|  | } |
|  | int main() |
|  | { |
|  | FILE \*fp; |
|  | int i,sc=0,wc=0; |
|  | int lf[26]={0},alpha[26]={0}; |
|  | char a,str[255]; |
|  | fp=fopen("ks.txt","r"); |
|  | fscanf(fp, "%[^\r]",str); |
|  | char \* a1 = strtok(str,"\n"); |
|  | while(a1!=NULL) |
|  | { |
|  | a1=To\_Lower(a1,lf); |
|  | sc+=symb\_count(a1); |
|  | wc+=wordify(a1); |
|  | a1=strtok(NULL,"\n"); |
|  | } |
|  | printf("1.Letter count = %d\n",count); |
|  | printf("2.Symbol count = %d\n",sc); |
|  | printf("3.Word Count = %d\n",wc); |
|  | letters\_not\_used(lf); |
|  | frequent\_letters(lf,alpha); |
|  | printf("5.Three Common letters are : %c,%c,%c\n",alpha[0],alpha[1],alpha[2]); |
|  | sort(); |
|  | if(wc>=3) |
|  | { |
|  | printf("6.Top three Common words : "); |
|  | p=head; |
|  | for(i=0;i<3;i++) |
|  | { |
|  | printf("%s, ",p->str); |
|  | p=p->next; |
|  | } |
|  | printf("\n"); |
|  | } |
|  | printf("7.No.of Words Occured once : %d\n",occured\_once()); |
|  | p=head; |
|  | fclose(fp); |
|  | return 0; |
|  | } |

2. Write a function that returns true if the given Binary Tree is SumTree else false.   
A SumTree is a Binary Tree where the value of a node is equal to sum of the nodes present in its left subtree and right subtree. An empty tree is SumTree and sum of an empty tree can be considered as 0. A leaf node is also considered as SumTree.   
  
Following is an example of SumTree.   
  
    26   
  /        \   
10        3   
/   \         \   
4   6         3

|  |
| --- |
| #include<stdio.h> |
|  |  |
|  | int sumTree(int i,int n,int \*tree) |
|  | { |
|  | if(i>=n) |
|  | { |
|  | return 0; |
|  | } |
|  | else |
|  | { |
|  | return tree[i] + sumTree(2\*i+1,n,tree) + sumTree(2\*i+2,n,tree); |
|  | } |
|  | } |
|  |  |
|  | int isSumTree(int i,int n,int \*tree) |
|  | { |
|  | if(i>=n) |
|  | { |
|  | return 1; |
|  | } |
|  | if(sumTree(2\*i+1,n,tree) + sumTree(2\*i+2,n,tree) == tree[i]) |
|  | { |
|  | return 1 \* sumTree(2\*i+1,n,tree) \* sumTree(2\*i+2,n,tree); |
|  | } |
|  | else |
|  | { |
|  | return 0; |
|  | } |
|  | } |
|  |  |
|  | int main() |
|  | { |
|  | int i,n; |
|  | int tree[10]; |
|  | printf("Enter number of nodes: "); |
|  | scanf("%d",&n); |
|  | printf("Enter the inorder traversal: "); |
|  | for(i=0;i<n;i++) |
|  | { |
|  | scanf("%d",&tree[i]); |
|  | } |
|  | if(isSumTree(0,n,tree)) |
|  | { |
|  | printf("The tree is a sum tree"); |
|  | } |
|  | else |
|  | { |
|  | printf("The tree is not a sum tree"); |
|  | } |
|  |  |
|  | } |

3. Find sum of all left leaves in a given Binary Tree.

|  |
| --- |
| #include <stdio.h> |
|  |  |
|  |  |
|  | #include<stdio.h> |
|  |  |
|  | struct Node |
|  | { |
|  | int value; |
|  | struct Node\* left,\*right; |
|  | }; |
|  |  |
|  | struct Node\* newNode(int x) |
|  | { |
|  | struct Node\* node = (struct Node \*)malloc(sizeof(struct Node)); |
|  | node->value = x; |
|  | node->left = NULL; |
|  | node->right=NULL; |
|  | return node; |
|  | } |
|  |  |
|  | int isLeaf(struct Node\* head) |
|  | { |
|  | if(head!=NULL && head->left == NULL && head->right == NULL) |
|  | { |
|  | return 1; |
|  | } |
|  | else |
|  | { |
|  | return 0; |
|  | } |
|  | } |
|  |  |
|  | int leftSum(struct Node\* head) |
|  | { |
|  | int sum=0; |
|  | if(head == NULL) |
|  | { |
|  | return 0; |
|  | } |
|  | if(isLeaf(head->left)) |
|  | { |
|  | sum = head->left->value; |
|  | } |
|  | return sum + leftSum(head->left)+ leftSum(head->right); |
|  | } |
|  |  |
|  | int main() |
|  | { |
|  | struct Node \*root=NULL; |
|  | root = newNode(10); |
|  | root->left = newNode(1); |
|  | root->right = newNode(10); |
|  |  |
|  | root->left->right = newNode(4); |
|  | root->left->left = newNode(3); |
|  |  |
|  | root->right->left = newNode(15); |
|  | printf("The sum is %d",leftSum(root)); |
|  |  |
|  | } |



4. A business application represents its customers data in JSON format (as shown in the image).   
  
Design a proper C structure to store the data.   
  
Share only the C structure. No program required.

|  |
| --- |
| Typedef char\* string; |
|  |  |
|  | struct contactPerson{ |
|  | string contact\_person\_id; |
|  | string first\_name; |
|  | string last\_name; |
|  | string email; |
|  | }; |
|  |  |
|  | struct contact{ |
|  | string contact\_id; |
|  | string contact\_name; |
|  | string contact\_street; |
|  | string area; |
|  | string city; |
|  | struct contactPerson \*contactPersons; |
|  | } \*contacts; |

5. You will be provided with set of Student ID (integer) and the Name. You have to list out the First name of the students (strip out the initials and the last/surname) along with the corresponding count and the array of IDs.   
  
Design proper C structure for it and execute the task.   
  
Sample Input   
  
1 B.Raghav   
2  Ashok V   
3  Raghav Bala   
4  Suresh Ramalingam   
5  KM.Suresh   
  
O/P   
  
Raghav, 2,  [1,3]   
Ashok, 1, [2]   
Suresh, 2, [4,5]

|  |
| --- |
| #include <stdio.h> |
|  | #include<stdlib.h> |
|  | #include<string.h> |
|  |  |
|  | struct student |
|  | { |
|  | int id,count,l; |
|  | int index[30]; |
|  | char \*name,\*firstname; |
|  | }; |
|  |  |
|  | void separatingFirstName(char \*name,char \*firstname) |
|  | { |
|  | int l=0,i; |
|  | for(i=0;name[i]!='\0';i++) |
|  | { |
|  | if(name[i]==' ') |
|  | { |
|  | break; |
|  | } |
|  | else if(name[i]=='.') |
|  | { |
|  | l=0; |
|  | continue; |
|  | } |
|  | firstname[l]=name[i]; |
|  | l++; |
|  | } |
|  | firstname[l]='\0'; |
|  | } |
|  |  |
|  | void findingduplicate(struct student \*s,int n) |
|  | { |
|  | int i,j,l=0; |
|  | for(i=0;i<n;i++) |
|  | { |
|  | if(s[i].firstname==NULL) |
|  | { |
|  | continue; |
|  | } |
|  | l=0; |
|  | s[i].count=1; |
|  | s[i].index[l]=i+1; |
|  | l++; |
|  | for(j=i+1;j<n;j++) |
|  | { |
|  | if(strcmp(s[i].firstname,s[j].firstname)==0) |
|  | { |
|  | s[i].index[l]=j+1; |
|  | l++; |
|  | s[i].count++; |
|  | s[j].firstname=NULL; |
|  | } |
|  | } |
|  | s[i].l=l; |
|  | } |
|  | } |
|  |  |
|  |  |
|  | int main(void) |
|  | { |
|  | int numStud,i,j; |
|  | struct student \*stud; |
|  | printf("Enter the number of student\n"); |
|  | scanf("%d",&numStud); |
|  | stud=(struct student\*)malloc(numStud\*sizeof(struct student)); |
|  |  |
|  | for(i=0;i<numStud;i++) |
|  | { |
|  | stud[i].name=(char\*)malloc(30\*sizeof(char)); |
|  | scanf("%d %[^\n]",&stud[i].id,stud[i].name); |
|  | } |
|  |  |
|  | for(i=0;i<numStud;i++) |
|  | { |
|  | stud[i].firstname=(char\*)malloc(30\*sizeof(char)); |
|  | separatingFirstName(stud[i].name,stud[i].firstname); |
|  | } |
|  |  |
|  | findingduplicate(stud,numStud); |
|  |  |
|  | for(i=0;i<numStud;i++) |
|  | { |
|  | if(stud[i].firstname !=NULL) |
|  | { |
|  | printf("%s,%d,[",stud[i].firstname,stud[i].count); |
|  | for(j=0;j<stud[i].l;j++) |
|  | { |
|  | printf("%d",stud[i].index[j]); |
|  | if(stud[i].count!=1) |
|  | { |
|  | stud[i].count--; |
|  | printf(","); |
|  | } |
|  | } |
|  | printf("]\n"); |
|  | } |
|  | } |
|  | return 0; |
|  | } |

6. Function should parse a hex string, and turn it into an array of 8-bit integers.   
  
Input: A null-terminated string of valid hex digits, that will be up to 64 hex characters long.   
  
Output: An array of 8-bit integers, each representing 2 characters of hex data. Should be at least half the length of the incoming string.   
  
void hex\_to\_bytearray (const char\* hex\_in, unsigned char\* int\_out){   
}   
  
Function should return a pointer to an array of 4, 8-bit integers corresponding to the values contained in the incoming 32-bit int.   
  
Example: input is 0x12345678, output should be a pointer to {0x12, 0x34, 0x56, 0x78} If possible, a typedef or a [#define](https://connect.zoho.com/portal/intranet/#tags/define) should be used instead of a function call.   
  
unsigned char\* int\_to\_array(unsigned int big\_int){   
}

7. Addition of two single linked list and store a result in third single linked list   
  
Note:   
  
1. Don't reverse the linked list..   
2. In case of diffrent lenght of input linked list don't make them of same lenght for calculation

|  |
| --- |
| #include<stdio.h> |
|  |  |
|  | struct Node |
|  | { |
|  | int x; |
|  | struct Node \*next; |
|  | }; |
|  |  |
|  |  |
|  | struct Node\* insert(int x,struct Node \*head) |
|  | { |
|  | struct Node \*p = (struct Node\*)malloc(sizeof(struct Node)); |
|  | p->x=x; |
|  | p->next=head; |
|  | return p; |
|  | } |
|  |  |
|  | struct Node\* reverse(struct Node \*q) |
|  | { |
|  | struct Node \*p,\*temp; |
|  | p=NULL; |
|  | while(q!=NULL) |
|  | { |
|  | temp = q->next; |
|  | q->next = p; |
|  | p=q; |
|  | q=temp; |
|  | } |
|  | return p; |
|  | } |
|  |  |
|  | struct Node\* add(struct Node\* p,struct Node\* q) |
|  | { |
|  | int carry=0,sum; |
|  | struct Node\* h3=NULL; |
|  |  |
|  | while(p!=NULL && q!=NULL) |
|  | { |
|  | sum = p->x + q->x + carry; |
|  | carry = (sum>9)?1:0; |
|  | sum = sum%10; |
|  |  |
|  | h3 = insert(sum,h3); |
|  | p=p->next; |
|  | q=q->next; |
|  | } |
|  |  |
|  | while(p!=NULL ) |
|  | { |
|  | sum = p->x + carry; |
|  | carry = (sum>9)?1:0; |
|  | sum = sum%10; |
|  |  |
|  | h3 = insert(sum,h3); |
|  | p=p->next; |
|  | } |
|  | while(q!=NULL) |
|  | { |
|  | sum = q->x + carry; |
|  | carry = (sum>9)?1:0; |
|  | sum = sum%10; |
|  |  |
|  | h3 = insert(sum,h3); |
|  | q=q->next; |
|  | } |
|  | if(carry) |
|  | { |
|  | h3=insert(1,h3); |
|  | } |
|  | return h3; |
|  | } |
|  |  |
|  |  |
|  | void display(struct Node\* p) |
|  | { |
|  | while(p!=NULL) |
|  | { |
|  | printf("%d->",p->x); |
|  | p=p->next; |
|  | } |
|  | printf("NULL\n"); |
|  | } |
|  |  |
|  | struct Node\* toLinklist(int n,struct Node\* head) |
|  | { |
|  | while(n>0) |
|  | { |
|  | head = insert(n%10,head); |
|  | n/=10; |
|  | } |
|  | return head; |
|  | } |
|  |  |
|  | int main() |
|  | { |
|  | struct Node \*h1=NULL,\*h2=NULL,\*h3=NULL; |
|  | int a,b; |
|  | printf("Enter two numbers: "); |
|  | scanf("%d%d",&a,&b); |
|  | h1=toLinklist(a,h1); |
|  | h2=toLinklist(b,h2); |
|  |  |
|  | printf("First num:\n"); |
|  | display(h1); |
|  | printf("Second num:\n"); |
|  | display(h2); |
|  |  |
|  | h1 = reverse(h1); |
|  | h2 = reverse(h2); |
|  | h3=add(h1,h2); |
|  |  |
|  | printf("Sum:\n"); |
|  | display(h3); |
|  |  |
|  |  |
|  | } |

8. For a given date in YYYY-MM-DD format, write a program with necessary functions to get the following information.   
  
Input: 2016-03-01   
  
i) The Year (2016)   
ii) The Previous Year (2015)   
iii) The Month in number (3)   
iv) The Month name (March)   
v) The Previous month (Feb)   
vi) The Day in number (1)   
vii) The Day name (Tuesday)   
viii) The Day index of the year (61) ie (31+29+1)   
ix) Date before N days [If N=2, then O/P 2016-02-28]   
x) Difference in days between 2 days.   
  
Conditions   
i) Use C-Structures.   
ii) Write functions for every task.  All function should return appropriate data. ie No print inside the function.   
  
[#calendar](https://connect.zoho.com/portal/intranet/#tags/calendar) [#utility-function](https://connect.zoho.com/portal/intranet/#tags/utility-function) [#c-structures](https://connect.zoho.com/portal/intranet/#tags/c-structures)

|  |
| --- |
| #include<stdio.h> |
|  | #include<string.h> |
|  | #include<math.h> |
|  | char \*month\_list[]={"Decmber","January","February","March","April","May","June","July","August","September","October","November","December"}; |
|  | char \*day\_list[]={"Saturday","Sunday","Monday","Tuesday","Wednesday","Thursday","Friday"}; |
|  | int day\_per\_month[13]={0,31,28,31,30,31,30,31,31,30,31,30,31}; |
|  | struct format |
|  | { |
|  | char str[12]; |
|  | int given[12]; |
|  | int year,month,date; |
|  | int leap,day\_index,new\_date; |
|  | }f,reference,new; |
|  | int find\_yr(struct format); |
|  | int find\_mnth(struct format); |
|  | int find\_day(struct format); |
|  | void before\_Ndays(struct format, struct format ,int); |
|  | int georgian(struct format ); |
|  | int find\_in\_between(struct format); |
|  | int check\_leap(int); |
|  | int find\_day\_index(struct format); |
|  | int find\_day\_name(struct format); |
|  | int main() |
|  | { |
|  | scanf("%s",f.str); |
|  | int i; |
|  | for(i=0;i<12;i++) |
|  | { |
|  | if(f.str[i]!='0') |
|  | { |
|  | f.given[i]=f.str[i]-'0'; |
|  | } |
|  | } |
|  | //to convert string input to integers |
|  | f.year=(((f.given[0])\*1000)+((f.given[1])\*100)+((f.given[2])\*10)+((f.given[3])\*1)); |
|  | f.month=(((f.given[5])\*10)+((f.given[6]))); |
|  | f.date=(((f.given[8]\*10))+f.given[9]); |
|  | //date validation |
|  | if((f.year<0)||(f.month>12)||(f.month<0)||(f.date>31)||(strlen(str)!=11)) |
|  | { |
|  | printf("Date is not valid"); |
|  | } |
|  | else |
|  | { |
|  | int yr\_no,month\_no,day\_no,index,new\_date,georgian,in\_between; |
|  | //calling various functions |
|  | yr\_no=find\_yr(f); |
|  | month\_no=find\_mnth(f); |
|  | day\_no=find\_day(f); |
|  | f.leap=check\_leap(f.year); |
|  | index=find\_day\_index(f); |
|  | before\_Ndays(f,new,index); |
|  | georgian=find\_day\_name(f); |
|  | in\_between=find\_in\_between(f); |
|  | //printing the results |
|  | printf("Current year is: %d\n ",yr\_no); |
|  | printf("Previous year is: %d\n",yr\_no-1); |
|  | printf("Month in number is: %d\n",month\_no); |
|  | printf("Month name is: %s\n",month\_list[month\_no]); |
|  | printf("Previous month name is:%s\n",month\_list[month\_no-1]); |
|  | printf("Date of the day is :%d\n",day\_no); |
|  | printf("Index of the day is: %d\n",index); |
|  | printf("Day for the given date is: %s\n",day\_list[georgian]); |
|  | printf("In between day count is %d\n",in\_between); |
|  | } |
|  | return 0; |
|  | } |
|  | int find\_yr(struct format f) |
|  | { |
|  | return (f.year); |
|  | } |
|  | int find\_mnth(struct format f) |
|  | { |
|  | return (f.month); |
|  | } |
|  | int find\_day(struct format f) |
|  | { |
|  | return (f.date); |
|  | } |
|  | int check\_leap(int yearcheck) |
|  | { |
|  | //check for leap year |
|  | f.leap=0; |
|  | if(yearcheck%100==0) |
|  | { |
|  | if((yearcheck%400)==0) |
|  | { |
|  | f.leap=1; |
|  | } |
|  | } |
|  | else if(yearcheck%4==0) |
|  | { |
|  | f.leap=1; |
|  | } |
|  | return (f.leap); |
|  | } |
|  | int find\_day\_index(struct format f) |
|  | { |
|  | int i; |
|  | if(f.month==1) |
|  | { |
|  | return f.date; |
|  | } |
|  | else |
|  | { |
|  | for(i=1;i<f.month;i++) |
|  | { |
|  | //counting the day index |
|  | f.day\_index=f.day\_index+day\_per\_month[i]; |
|  | } |
|  | if((f.leap==1)&&(f.month>2)) |
|  | { |
|  | //increment by 1 if leap year |
|  | f.day\_index++; |
|  | } |
|  | return f.day\_index+f.date; |
|  | } |
|  | } |
|  | int find\_day\_name(struct format f) |
|  | { |
|  | int h,q,K,J,temp\_month=f.month,temp\_year=f.year,m; |
|  | q=f.date; |
|  | if(f.month==1||(f.month==2)) |
|  | { |
|  | temp\_month=f.month+12; |
|  | temp\_year=f.year-1; |
|  | } |
|  | //finding the day using zeller's congurence method(wikipedia reference) |
|  | m=13\*(temp\_month+1)/5; |
|  | K=temp\_year%100; |
|  | J=temp\_year/100; |
|  | h=((q+m+K+(K/4)+(J/4)+(5\*J))%7); |
|  | return h; |
|  | } |
|  | void before\_Ndays(struct format f,struct format new,int index) |
|  | { |
|  | new=f; |
|  | int N,value; |
|  | scanf("%d",&N); |
|  | //calculating the date difference b/w given N and original date |
|  | while(N--) |
|  | { |
|  | new.date--; |
|  | } |
|  | value=new.date; |
|  | if (value<31) |
|  | { |
|  | value=1; |
|  | } |
|  | //processing the difference value to check year, month changes |
|  | if(value>365) |
|  | { |
|  | new.year=new.year-(new.year/365); |
|  | } |
|  | if(new.date<day\_per\_month[new.month]) |
|  | { |
|  | new.date=day\_per\_month[new.month-1]+new.date; |
|  | new.month=new.month-(value)%(day\_per\_month[new.month]); |
|  | } |
|  | printf("Date before N days is : %d-%d-%d\n",new.year,new.month,new.date); |
|  | } |
|  | int find\_in\_between(struct format f) |
|  | { |
|  | //getting a reference date |
|  | scanf("%d-%d-%d",&reference.year,&reference.month,&reference.date); |
|  | int ibdays=0,i; |
|  | int le,yeardays=0; |
|  | if(f.year==reference.year) |
|  | { |
|  | //counting no of days by using their index values |
|  | ibdays=((find\_day\_index(reference))-(find\_day\_index(f))); |
|  | } |
|  | else if(f.year-reference.year==1) |
|  | { |
|  | ibdays=((365-find\_day\_index(f))+(find\_day\_index(reference))); |
|  | } |
|  | else |
|  | { |
|  | //if the dates are from different years, code to consider that |
|  | for(i=reference.year;i<(f.year);i++) |
|  | { |
|  | le=check\_leap(i); |
|  | if(le==1) |
|  | { |
|  | yeardays=yeardays+366; |
|  | } |
|  | else |
|  | { |
|  | yeardays=yeardays+365; |
|  | } |
|  |  |
|  | } |
|  | ibdays=((365-find\_day\_index(f))+(find\_day\_index(reference))); |
|  | ibdays=ibdays+yeardays; |
|  | } |
|  | if(ibdays<0) |
|  | { |
|  | ibdays=ibdays\*(-1); |
|  | } |
|  | return ibdays; |
|  | } |

9. Sort the elements in a given String 2D array.   
  
i) The String array contains only the numbers.   
ii) The array should be arranged based on numerical order.   
iii) String to number conversion is not allowed   
  
Ex:   
I/P   
Array: [ "1", "3", "11", "100", "19", "2", "5"]   
  
O/P   
Array: ["1", "2", "3", "5", "11", "19", "100"]

|  |
| --- |
| #include<stdio.h> |
|  | #include<string.h> |
|  |  |
|  | int compare(char a[],char b[]) |
|  | { |
|  | int i; |
|  | for(i=0;a[i]!='\0'&&a[i]==b[i];i++); |
|  | if(a[i]>b[i]) |
|  | return 1; |
|  | else |
|  | return 0; |
|  | } |
|  | int main() |
|  | { |
|  | char a[10][10]; |
|  | int i=0,j,n; |
|  |  |
|  | scanf("%d",&n); |
|  | for(i=0;i<n;i++) |
|  | { |
|  | scanf("%s",a[i]); |
|  | } |
|  |  |
|  | for(i=0;i<n;i++) |
|  | { |
|  | for(j=i+1;j<n;j++) |
|  | { |
|  | if(strlen(a[i])>=strlen(a[j])) |
|  | { |
|  | if(strlen(a[i]) == strlen(a[j])) |
|  | { |
|  | int flag = compare(a[i],a[j]); |
|  | if(flag == 0) |
|  | { |
|  | continue; |
|  | } |
|  | } |
|  | char temp[10]; |
|  | strcpy(temp,a[i]); |
|  | strcpy(a[i],a[j]); |
|  | strcpy(a[j],temp); |
|  |  |
|  | } |
|  | } |
|  | } |
|  |  |
|  | printf("Sorted array: \n"); |
|  | for(i=0;i<n;i++) |
|  | { |
|  | printf("%s\t",a[i]); |
|  | } |
|  | } |

10. A file marks.csv that consists the Student name, roll no and the subject marks (Maths, Physics, Chemistry, Lang1 & Lang2) was given to the clerk. The clerk need to rank the students for counseling (TNEA??) based on the total and results. He also need to filter and  sort the student records by rank and result and store it into a file named rank.csv.   
  
Write a program to help the clerk.   
  
Conditions   
  
i)  Assume there are totally 15 records   
ii) The max mark for each subject is 200   
iii) Those who got scored >=35% on each subject will be treated as PASS.   
iv) Apply BE counseling aggregate formula (M/2)+((P+C)/4)   
v) If more than one student got the same aggregate, then weight-age will be given to Total. The next preference will be given to LANG1   
  
[#file-access](https://connect.zoho.com/portal/intranet/#tags/file-access) [#c-structures](https://connect.zoho.com/portal/intranet/#tags/c-structures) [#effective-comparison](https://connect.zoho.com/portal/intranet/#tags/effective-comparison) [#sorting](https://connect.zoho.com/portal/intranet/#tags/sorting)

|  |
| --- |
| #include<stdio.h> |
|  | #include<stdlib.h> |
|  | #include<string.h> |
|  | struct str |
|  | { |
|  | int mar[5]; |
|  | char name[25]; |
|  | int total; |
|  | int pf; |
|  | float cutoff; |
|  | }\*s; |
|  | int main() |
|  | { |
|  | FILE \*f; |
|  | FILE \*rank; |
|  | struct str temp; |
|  | char tStr[100]; |
|  | int t=1; |
|  | int number\_of\_lines=1; |
|  | char single; |
|  | rank=fopen("/Users/bharadwaj/Desktop/c/rank.csv","w"); |
|  | f=fopen("/Users/bharadwaj/Desktop/c/names.csv","r"); |
|  | if(f==NULL) |
|  | { |
|  | printf("Cant detect "); |
|  | return 0; |
|  | } |
|  | while((single=fgetc(f))!=EOF) |
|  | { |
|  | if(single=='\n') |
|  | { |
|  | number\_of\_lines++; |
|  | } |
|  | } |
|  | rewind(f); |
|  | s=(struct str\*)malloc(sizeof(struct str)\*number\_of\_lines); |
|  |  |
|  | for(int k=0;k<number\_of\_lines;k++) |
|  | { |
|  | fscanf(f,"%[^,],%d,%d,%d,%d,%d",s[k].name,&s[k].mar[0],&s[k].mar[1],&s[k].mar[2],&s[k].mar[3],&s[k].mar[4]); |
|  | s[k].total=s[k].mar[0]+s[k].mar[1]+s[k].mar[2]+s[k].mar[3]+s[k].mar[4]; |
|  | s[k].cutoff=((s[k].mar[0])/2)+ (s[k].mar[1]+s[k].mar[2])/4; |
|  | } |
|  | for(int i=0;i<number\_of\_lines;i++) |
|  | { |
|  | for(int j=i+1;j<number\_of\_lines;j++) |
|  | { |
|  | if(s[i].cutoff<s[j].cutoff) |
|  | { |
|  | temp=s[i]; |
|  | s[i]=s[j]; |
|  | s[j]=temp; |
|  | } |
|  | else if(s[i].cutoff==s[j].cutoff) |
|  | { |
|  | if(s[i].total<s[j].total) |
|  | { |
|  | temp=s[i]; |
|  | s[i]=s[j]; |
|  | s[j]=temp; |
|  | }else if(s[i].total==s[j].total) |
|  | { |
|  | if(s[i].mar[3]<s[j].mar[3]) |
|  | { |
|  | temp=s[i]; |
|  | s[i]=s[j]; |
|  | s[j]=temp; |
|  | } |
|  | } |
|  | } |
|  | } |
|  | } |
|  | for(int i=0;i<number\_of\_lines;i++) |
|  | { |
|  | for(int j=0;j<5;j++) |
|  | { |
|  | if(s[i].mar[j]<70) |
|  | { |
|  | s[i].pf=0; |
|  | } |
|  | else |
|  | { |
|  | s[i].pf=1; |
|  | } |
|  | } |
|  | } |
|  | for(int i=0;i<number\_of\_lines;i++) |
|  | { |
|  | if(s[i].pf) |
|  | { |
|  | fprintf(rank, "%d,%s,%0.2f,Pass\n",t,s[i].name,s[i].cutoff); |
|  | t++; |
|  | } |
|  | } |
|  | return 0; |
|  | } |

11. For a given number N, find the next immediate palindrome number   
  
I/p: 808   
O/p: 818   
  
I/P: 2133   
O/P: 2222

|  |
| --- |
| #include <stdio.h> |
|  | #include <math.h> |
|  | int digitize(int n){ |
|  | int i=0,q; |
|  | while( (q = n/10) > 0){ |
|  | i++; |
|  | n = n/10; |
|  | } |
|  |  |
|  | return i+1; |
|  | } |
|  | int nxtPalin(int n){ |
|  | int i,p=0,palin=n,pow\_10,pf=0,rem,prev; |
|  | int power=1,prev\_power=1; |
|  |  |
|  | while(pf != 1){ |
|  | pow\_10 = digitize(palin); |
|  | int a[pow\_10+1]; |
|  | prev = 0; |
|  | power=1;prev\_power=1; |
|  | for(i=1;i<=pow\_10;i++){ |
|  | prev\_power=power; |
|  | power \*= 10; |
|  | rem = palin % power; |
|  | //printf("%d\n",power ); |
|  | a[i-1] = (rem - prev) / prev\_power; |
|  | prev = rem; |
|  | } |
|  | for(i=0;i<pow\_10/2;i++){ |
|  | if(a[i] != a[pow\_10 -1 -i]){ |
|  | p=1; |
|  | } |
|  | } |
|  | if( p == 0) { |
|  | pf = 1; |
|  | break; |
|  | }else{ |
|  | p=0; |
|  | palin++; |
|  | } |
|  | } |
|  |  |
|  | return palin; |
|  |  |
|  | } |
|  |  |
|  | int main(){ |
|  | int n,next; |
|  | printf("Enter the number: "); |
|  | scanf("%d",&n); |
|  | next=nxtPalin(n+1); |
|  | printf("The next palindrome is %d\n", next ); |
|  | return 0; |
|  | } |

12. Convert the given number (0-999) into its word representation   
  
I/p: 123   
O/p: One hundred and twenty three   
  
I/p: 19   
O/p: Nineteen

|  |
| --- |
| #include<stdio.h> |
|  |  |
|  | void main() |
|  | { |
|  | char ones[][10]={"one","two","three","four","five","six","seven","eight","nine"}; |
|  | char tens[][10]={"twenty","thirty","forty","fifty","sixty","seventy","eighty","ninety"}; |
|  | char teens[][10]={"eleven","twelve","thirteen","fourteen","fifteen","sixteen","seventeen","eighteen","nineteen"}; |
|  | int n,digit[3]={-1,-1,-1},i=0; |
|  |  |
|  | printf("Enter n: "); |
|  | scanf("%d",&n); |
|  |  |
|  | while(n>0) |
|  | { |
|  | digit[i++]=n%10; |
|  | n=n/10; |
|  | } |
|  |  |
|  | //hundreds place |
|  | if(digit[2] != -1) |
|  | { |
|  | printf("%s hundred",ones[digit[2]-1]); |
|  | if(digit[1] != 0 || digit [0] != 0) |
|  | { |
|  | printf(" and "); |
|  | } |
|  | else |
|  | { |
|  | return; |
|  | } |
|  | } |
|  |  |
|  | //tens place |
|  | if(digit[1] != -1 && digit[1] != 0) |
|  | { |
|  |  |
|  | if(digit[1] != 1) |
|  | { |
|  | printf("%s ",tens[digit[1]-2]); |
|  | if(digit[0]!= 0) |
|  | { |
|  | printf("%s ",ones[digit[0]-1]); |
|  | } |
|  | } |
|  | else |
|  | { |
|  | if(digit[0] == 0) |
|  | { |
|  | printf("ten "); |
|  | } |
|  | else |
|  | printf("%s ",teens[digit[0]-1]); |
|  | } |
|  | } |
|  | else |
|  | if(digit[0]!=0) //ones place |
|  | { |
|  | if(digit[0]!= -1 ) |
|  | { |
|  | printf("%s",ones[digit[0]-1]); |
|  | } |
|  | else |
|  | { |
|  | printf("%s","zero"); |
|  | } |
|  |  |
|  | } |
|  | } |

13. Write a program to find the  longest sub-string and length without repeating characters in a given string.   
  
Ex:   s = “abcabc”, the longest sub-string without repeating characters is “abc” and it's the length is 3.

|  |
| --- |
| #include<stdio.h> |
|  |  |
|  | char\* findSubstring(char s[],int n,char substring[]) |
|  | { |
|  | int start=0,end,maxlen=1,curlen=1,max\_start=0,max\_end=0; |
|  | int past\_index,i; |
|  | int visited[26]; |
|  |  |
|  | for(i=0;i<26;i++) |
|  | visited[i]=-1; |
|  |  |
|  | visited[s[start]-'a'] = 0; |
|  |  |
|  | for(end=1;end<n;end++) |
|  | { |
|  | past\_index = visited[s[end] - 'a']; |
|  | if( past\_index == -1 || start > past\_index) |
|  | { |
|  | curlen++; |
|  | } |
|  | else |
|  | { |
|  | if(curlen>maxlen) |
|  | { |
|  | maxlen=curlen; |
|  | max\_start = start; |
|  | max\_end = end; |
|  | } |
|  | curlen = end - past\_index; |
|  | start = past\_index + 1; |
|  | } |
|  |  |
|  | visited[s[end] - 'a'] = end; |
|  | } |
|  |  |
|  | if(curlen>maxlen) |
|  | { |
|  | maxlen=curlen; |
|  | max\_start = start; |
|  | max\_end = end; |
|  | } |
|  |  |
|  | for(i=0;i<maxlen;i++) |
|  | { |
|  | substring[i] = s[max\_start++]; |
|  | } |
|  |  |
|  | substring[i] = '\0'; |
|  |  |
|  | return maxlen; |
|  |  |
|  | } |
|  |  |
|  | int main() |
|  | { |
|  | char \*s,\*substring; |
|  | int len,n; |
|  | printf("Enter length of string: "); |
|  | scanf("%d",&n); |
|  |  |
|  | s = (char\* )malloc(sizeof(char)\*n); |
|  | substring = (char\*)malloc(sizeof(char)\*n); |
|  |  |
|  | printf("Enter string: "); |
|  | scanf("%s",s); |
|  |  |
|  | len = findSubstring(s,n,substring); |
|  | printf("Longest substring is: %s with length: %d \n",substring,len); |
|  | } |

14. Write a program to find the SquareRoot of a given number using Babylonian method.

|  |
| --- |
| #include<stdio.h> |
|  |  |
|  | int main() |
|  | { |
|  | int n,num,c=10; |
|  | float ans, guess=1, r; |
|  | printf("\n Enter the number"); |
|  | scanf("%d",&num); |
|  | n=num; |
|  | r = n/guess; |
|  | guess = (guess + r)/2; |
|  | while (c > 0) |
|  | { |
|  | r = n/guess; |
|  | guess = (guess + r)/2; |
|  | if (guess <= (guess \* 0.01) + guess) |
|  | { |
|  | ans = guess; |
|  | } |
|  | else |
|  | { |
|  | r = n/guess; |
|  | } |
|  | guess = (guess + r)/2; |
|  | c-=1; |
|  | } |
|  | printf(" The square root is %f",ans); |
|  | return 0; |
|  | } |

15. Write a Program to find the next prime number of a given number.   
  
Ex:   
I/P : 8   
O/P: 11   
  
I/P: 121   
O/P: 127

|  |
| --- |
| #include<math.h> |
|  | int prime(int n); |
|  | int main() |
|  | { |
|  | int n,i,flag,temp,val=1; |
|  | scanf("%d",&n); |
|  | if(n==1) |
|  | { |
|  | printf("Nearest prime : %d\n",2); |
|  | } |
|  | else |
|  | { |
|  | temp=n; |
|  | for(i=n+1;val==1;i++) |
|  | { |
|  | val = prime(i); |
|  | } |
|  | printf("Nearest prime is : %d\n",i-1); |
|  |  |
|  | } |
|  | return 0; |
|  | } |
|  | int prime(int n) |
|  | { |
|  | int i,flag=0,cond; |
|  | cond=sqrt(n); |
|  | for(i=2;i<=cond;i++) |
|  | { |
|  | if(n%i==0) |
|  | { |
|  | flag++; |
|  | break; |
|  | } |
|  | } |
|  | return flag; |
|  |  |
|  | } |

16.

char ch=0;   
int i=0;   
while(++ch)   
{   
   i++;   
}   
printf("%d %d\n", i, ch);   
  
Would you get o/p for the above program?   
  
If yes, what is the output? How did you got that?   
If no,  why?   
  
[#titbits](https://connect.zoho.com/portal/intranet/#tags/titbits)

Output is : 255 0   
Coz, the value ch goes from 1 to 127 then -128 to 0 in while loop    
and so while loop executes 255 times and i is set to 255

17.

int main()   
{   
   int x=20;   
   {   
      <YOUR STATEMENT HERE>   
      printf("%d ", x);   
      x++;   
   }   
   x++;   
   printf("%d\n", x);   
   return 0;   
}   
  
What statement you would write in the above snippet to get the output   
  
i) 10 12   
  
and what statement for output   
  
ii) 10 21   
  
[#titbits](https://connect.zoho.com/portal/intranet/#tags/titbits)

To get i) as output - the statement is   
x =10;   
This changes x value to 10.   
To get ii) as output - the statement is   
int x =10;   
This initializes a new variable with value 10 visible only within the inner block

18. What would be the o/p of this snippet?   
  
printf("%02d\n", printf("%d", 100));   
  
write the explanation too.

o/p will be 10003. because the printf statement starts execution fromright to left. so printf("%d",100) gets executed and for the next printf we get 3 since the length of 100 is 3. as it is "%02d" we get 03

int x=10, y=20;;   
;   
printf("%d", x);   
  
Would you able to compile it??   
  
If No, which statement is wrong??   
If Yes, why is that so?

; is null statement continuous ;; doesn't cause any error so the output is generated

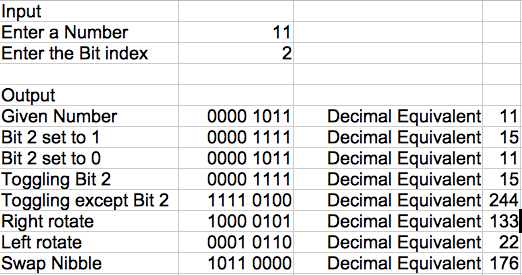
if(X)   
{   
    printf("Enrich");   
}   
else   
{   
   printf("Zoho");   
}   
What  statement/condition you would replace "X" to get the output as   
  
Zoho Enrich   
  
Don't share the link, just comment here the condition alone.

if(printf("Zoho "))   
{   
      printf("Enrich");   
}   
else   
{   
      printf("Zoho");   
}   
  
the if condition is satisfied because printf("Zoho") is a true value. hence it prints Zoho Enrich

int i=10;   
for(i=1; i<=10;i++);   
{   
   printf("%d\n", i);   
}   
  
What would be the output?

Since the Control gets out of loop once i becomes 11 & 'i' value will be 11

19. Write a program to accomplish the following tasks   
  
1) Find the value of n-th bit of the given number (7<=n<=0)   
2) Set the n-th bit to 1 of the given number   
3) Unset the n-th bit (set it to 0) of the given number   
4) Toggle the n-th bit of the given number   
5) Toggle all the bit except n-th bit   
6) Right rotate a given numbet   
7) Left rotate a given number   
8) Swap the nibbles of a given byte   
  
Points to be Noted   
\* Use char and play with single byte for better understanding   
\* All operations need to be done on the given number   
\* Use separate functions for all the tasks   
\* Arrays should not be used   
\* Task functions should return the data. printing should be done in main function. No print inside task functions



|  |
| --- |
| #include <stdio.h> |
|  | void printF(int ); |
|  | void bittog(int ,int ); |
|  | void bittogex(int ,int ); |
|  | void bitset(int ,int); |
|  | void bitz(int ,int); |
|  | void swap(int); |
|  | void bitriR(int); |
|  | void bitriL(int); |
|  | int main(int argc, const char \* argv[]) { |
|  | int num=11; |
|  | printf("given bit \t"); |
|  | printF(num); |
|  | int k=2; |
|  | bitset(num,k); |
|  | bitz(num,k); |
|  | bittog(num,k); |
|  | bittogex(num,k); |
|  | bitriR(num); |
|  | bitriL(num); |
|  | swap(num); |
|  | } |
|  | void swap(int num) |
|  | { |
|  | printf("After swapping nibble"); |
|  | printF(num<<4|num>>4); |
|  | } |
|  | void bitriR(int num) |
|  | { |
|  | printf("After right rotate"); |
|  | num=num>>1|num<<7; |
|  | printF(num); |
|  | } |
|  | void bitriL(int num) |
|  | { |
|  | printf("After left rotate"); |
|  | num=num<<1|num>>7; |
|  | printF(num); |
|  | } |
|  | void bittogex(int num,int k) |
|  | { |
|  | num=(~num)|(1<<k); |
|  | printf("After toggling other than bit %d\t",k); |
|  | printF(num); |
|  | } |
|  | void bittog(int num,int k) |
|  | { |
|  | num=num^(1<<k); |
|  | printf("After toggling bit %d\t",k); |
|  | printF(num); |
|  | } |
|  |  |
|  |  |
|  | void bitz(int num,int k) |
|  | { |
|  | num=num&(~(1<<k)); |
|  | printf("Bit %d to 0\t",k); |
|  | printF(num); |
|  | } |
|  |  |
|  |  |
|  | void bitset(int num,int k) |
|  | { |
|  | num=num|1<<k; |
|  | printf("Bit set %d to 1 \t",k); |
|  | printF(num); |
|  | } |
|  |  |
|  |  |
|  | void printF(int num) |
|  | { |
|  | int n=128; |
|  | while(n>0) |
|  | { |
|  | if((num&n)==0) |
|  | { |
|  | printf("0"); |
|  | } |
|  | else |
|  | { |
|  | printf("1"); |
|  | } |
|  | n=n>>1; |
|  | } |
|  | printf("\n"); |
|  | } |

20. Write a program to represent a given decimal number in requested base.   
  
Use appropriate function signature.   
  
Ex 1:   
Input   
Number : 10   
Base : 2   
  
Output   
1010   
  
Ex2:   
  
Input   
Number : 10   
Base : 5   
  
Output   
20   
  
Ex 3:   
  
Input   
Number : 33   
Base : 17   
  
Output:   
1G

|  |
| --- |
| #include<stdio.h> |
|  |  |
|  | char\* toBase(int n,int b) |
|  | { |
|  | int res,i=0; |
|  | char \*stack = (char\*) malloc(sizeof(char)); |
|  |  |
|  | while(n>0) |
|  | { |
|  | res = n%b; |
|  |  |
|  | if(res>9) |
|  | { |
|  | stack[i++] = 'A' + (res-10); |
|  | } |
|  | else |
|  | { |
|  | stack[i++] = res + '0'; |
|  | } |
|  |  |
|  | n = n/b; |
|  | } |
|  |  |
|  | stack[i--] = '\0'; |
|  |  |
|  | //reversing the stack |
|  | int j=0; |
|  | char temp; |
|  |  |
|  | while(j<i) |
|  | { |
|  | temp = stack[i]; |
|  | stack[i] = stack[j]; |
|  | stack[j] = temp; |
|  | i--; |
|  | j++; |
|  | } |
|  | return stack; |
|  | } |
|  |  |
|  | int main() |
|  | { |
|  | int num,base; |
|  | char \*result; |
|  | printf("Enter number: "); |
|  | scanf("%d",&num); |
|  | printf("Enter base: "); |
|  | scanf("%d",&base); |
|  |  |
|  | if(base>36 || base<2) |
|  | { |
|  | printf("The base must be between 2 and 36 \n"); |
|  | return; |
|  | } |
|  |  |
|  | result = toBase(num,base); |
|  | printf("Result: %s",result); |
|  | } |

21. Print the number of bits required to represent a decimal number   
  
Ex:   
Input: 5   
output : 3   
  
as decimal 5 will be represent as 101 in binary form   
  
Ex:   
Input 9   
Output: 4

|  |
| --- |
| #include <stdio.h> |
|  | int main(void) { |
|  | int number,n=2,count=1; |
|  | printf("enter the decimal number: \n"); |
|  | scanf("%d",&number); |
|  | while(number>=n) |
|  | { |
|  | count=count+1; |
|  | n=n\*2; |
|  | } |
|  | printf("the number of bits needed is : %d\n",count); |
|  | // your code goes here |
|  | return 0; |
|  | } |

22. Reverse a given number   
  
Input: 207   
Output : 702   
  
Input: 123   
Output: 321

|  |
| --- |
| #include <stdio.h> |
|  | void main() |
|  | { |
|  | int n,rev=0,s; |
|  | printf("Enter the number"); |
|  | scanf("%d", &n); |
|  | while(n!=0) |
|  | { |
|  | s=n%10; |
|  | rev=rev\*10; |
|  | rev=rev+s; |
|  | n=n/10; |
|  | } |
|  | printf("OUTPUT= %d",rev); |
|  | } |

23. List the subsets of the given array. No RECURSION Please.   
  
Input : {1,2,3}   
  
Output:   
1,   
1,2   
1,2,3   
1,3   
2,   
2,3   
3   
  
PS: The output can be listed in any order. It not needed to be in the same order as above.

|  |
| --- |
| #include<stdio.h> |
|  | #include<stdlib.h> |
|  | #include<conio.h> |
|  |  |
|  |  |
|  | void main() |
|  | { |
|  |  |
|  | int size,i,j,subset; |
|  | int \*arr; |
|  | clrscr(); |
|  | printf("Enter the size: "); |
|  | scanf("%d",&size); |
|  |  |
|  | arr=(int\*)malloc(size\* sizeof(int)); |
|  |  |
|  | printf("\nEnter the numbers: "); |
|  | for (i=0;i<size;i++) |
|  | { |
|  | scanf("%d",&arr[i]); |
|  | } |
|  | subset=1<<size; |
|  |  |
|  | for(i=1;i<subset;i++) |
|  | { |
|  | for(j=0;j<size && j<i;j++) |
|  | { |
|  | if(i&(1<<j)) |
|  | { |
|  | printf("%d ",arr[j]); |
|  | } |
|  | } |
|  | printf("\n"); |
|  | } |
|  | getch(); |
|  | } |

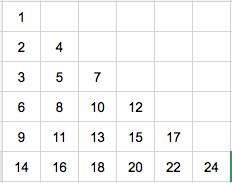
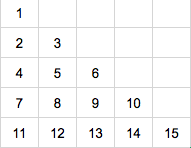
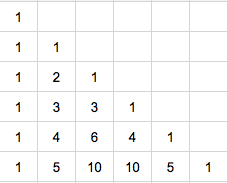
24. Construct the Slide Square Game   
Ref : <http://mypuzzle.org/sliding>   
  
\* Construct the square with hard coded random numbers [ No need to go dynamic number arrangements. we can practice it later ]   
\* Draw the square in the screen and accept keys [ U,D,L,R,Q]   
\* Adjust the square based on the key press   
\* Print Completed, when all the numbers are sorted

|  |
| --- |
|  |
| #include<stdio.h> |
|  | #include<stdlib.h> |
|  | #include<conio.h> |
|  | int zrow,zcol,row,col,i,j; |
|  | void printArray(int arr[3][3]); |
|  | int initialiseArr(int a[3][3]); |
|  | int swap(int arr[3][3],int row,int col); |
|  | int solved(int a[3][3]); |
|  | int main() |
|  | { |
|  | int arr[3][3]={2,5,8,1,3,6,8,7,9}; |
|  | char move; |
|  | arr[3][3]=initialiseArr(arr); |
|  | clrscr(); |
|  | do |
|  | { |
|  | system("cls"); |
|  | printArray(arr); |
|  | printf("\nEnter the move: U->Up,D->Down,R->Right,L->Left\t"); |
|  | scanf("%c",&move); |
|  | move=toupper(move); |
|  | switch(move) |
|  | { |
|  | case'U': |
|  | row=zrow-1; |
|  | col=zcol; |
|  | break; |
|  | case'D': |
|  | row=zrow+1; |
|  | col=zcol; |
|  | break; |
|  | case'R': |
|  | row=zrow; |
|  | col=zcol+1; |
|  | break; |
|  | case'L': |
|  | row=zrow; |
|  | col=zcol-1; |
|  | break; |
|  | case'Q': |
|  | printf("\nYou Quit"); |
|  | break; |
|  | default: |
|  | printf("\nEnter the correct move"); |
|  | } |
|  | arr[3][3]=swap(arr,row,col); |
|  | }while(move!='Q'); |
|  | solved(arr); |
|  | getch(); |
|  | return 0; |
|  | } |
|  | int initialiseArr(int a[3][3]) |
|  | { |
|  | a[2][2]=0; |
|  | zrow=zcol=2; |
|  | return a[3][3]; |
|  | } |
|  | void printArray(int arr[3][3]) |
|  | { |
|  | for(i=0;i<3;i++) |
|  | { |
|  | printf("\n-------\n"); |
|  | for(j=0;j<3;j++) |
|  | { |
|  | printf("|%d",arr[i][j]); |
|  | } |
|  | printf("|"); |
|  | } |
|  | } |
|  | int swap(int arr[3][3],int row,int col) |
|  | { |
|  | int temp; |
|  | temp=arr[row][col]; |
|  | arr[row][col]=arr[zrow][zcol]; |
|  | arr[zrow][zcol]=temp; |
|  | zrow=row; |
|  | zcol=col; |
|  | return arr[3][3]; |
|  | } |
|  | int solved(int arr[3][3]) |
|  | { |
|  | int k=1; |
|  | for(i=0;i<3;i++) |
|  | { |
|  | for(j=0;j<3;j++) |
|  | { |
|  | if(arr[i][j]==k &&arr[3][3]==0) |
|  | { |
|  | k++; |
|  | } |
|  | } |
|  | } |
|  | if(k==9) |
|  | { |
|  | printf("Puzzle solved You Win.."); |
|  | } |
|  | else |
|  | { |
|  | printf("Try once again..."); |
|  | } |
|  | return; |
|  | } |
|  |  |

25. Construct a Magic Square by Siamese method

|  |
| --- |
| #include <stdio.h> |
|  | void magicsq(int, int [][10]); |
|  | int main( ) |
|  | { |
|  | int size; |
|  | int a[10][10]; |
|  | printf("Enter the size: "); |
|  | scanf("%d", &size); |
|  | if (size % 2 == 0) |
|  | { |
|  | printf("Magic square works for an odd numbered size\n"); |
|  | } |
|  | else |
|  | { |
|  | magicsq(size, a); |
|  | } |
|  |  |
|  | return 0; |
|  | } |
|  |  |
|  | void magicsq(int size, int a[][10]) |
|  | { |
|  | int sqr = size \* size; |
|  | int i = 0, j = size / 2, k; |
|  | for (k = 1; k <= sqr; ++k) |
|  | { |
|  | a[i][j] = k; |
|  | i--; |
|  | j++; |
|  | if (k % size == 0) |
|  | { |
|  | i += 2; |
|  | --j; |
|  | } |
|  | else |
|  | { |
|  | if (j == size) |
|  | { |
|  | j -= size; |
|  | } |
|  | else if (i < 0) |
|  | { |
|  | i += size; |
|  | } |
|  | } |
|  | } |
|  | // printing |
|  | for (i = 0; i < size; i++) |
|  | { |
|  | for (j = 0; j < size; j++) |
|  | { |
|  | printf("%d\t", a[i][j]); |
|  | } |
|  | printf("\n"); |
|  | } |
|  | printf("\n"); |
|  | } |

26. construct these number triangles



|  |
| --- |
| #include<stdio.h> |
|  |  |
|  | int binCoeff(int i, int j) |
|  | { |
|  | if(i==j) |
|  | return 1; |
|  | else |
|  | if(i==0 || j==0) |
|  | return 0; |
|  | else |
|  | return binCoeff(i-1,j)+binCoeff(i-1,j-1); |
|  | } |
|  |  |
|  | void floydTri(int n) |
|  | { |
|  | int x=1,i,j; |
|  | for(i=0;i<=n;i++) |
|  | { |
|  | for(j=0;j<i;j++) |
|  | { |
|  | printf(" %d ",x++); |
|  | } |
|  | printf("\n"); |
|  | } |
|  | } |
|  |  |
|  | void oddEven(int n) |
|  | { |
|  | int odd=1,even=2,i,j; |
|  | for(i=0;i<n;i++) |
|  | { |
|  | for(j=0;j<i;j++) |
|  | { |
|  | if((i%2) != 0) //odd row |
|  | { |
|  | printf(" %d ",odd); |
|  | odd+=2; |
|  | } |
|  | else //even row |
|  | { |
|  | printf(" %d ",even); |
|  | even+=2; |
|  | } |
|  | } |
|  | printf("\n"); |
|  | } |
|  |  |
|  | } |
|  |  |
|  | void pascalTri(int n) |
|  | { |
|  | int i,j; |
|  | for(i=0;i<n;i++) |
|  | { |
|  | for(j=0;j<i;j++) |
|  | { |
|  | printf(" %d ",binCoeff(i,j+1)); |
|  | } |
|  | printf("\n"); |
|  | } |
|  |  |
|  | } |
|  |  |
|  | int main() |
|  | { |
|  | int n; |
|  | printf("Enter the number of rows: "); |
|  | scanf("%d",&n); |
|  |  |
|  | printf("\n Floyd's Triangle: \n "); |
|  | floydTri(n); |
|  |  |
|  | printf("\n OddEven Triangle: \n "); |
|  | oddEven(n); |
|  |  |
|  | printf("\n Pascal Triangle: \n "); |
|  | pascalTri(n); |
|  | } |