15B17CI371–Data Structures Lab ODD 2024

Week 5-LAB B

Practice Lab

* 1. **Write a program to find whether the number is Palindrome or not. A number is a Palindrome if it remains the same when its digits are reversed. Assumption: N is a positive integer.**

#include <iostream>

using namespace std;

int reversenum(int num,int temp)

{

if(num==0)

return temp;

temp=(temp\*10)+(num%10);

return reversenum(num/10,temp);

}

bool ispalindrome(int num)

{

int reversednum=reversenum(num,0);

return(num==reversednum);

}

int main()

{

int num;

cout<<"Enter a number: ";

cin>>num;

if(ispalindrome(num))

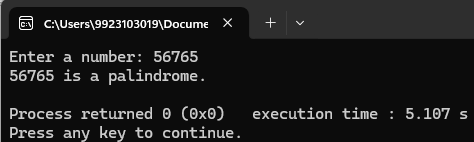
cout<<num<<" is a palindrome."<<endl;

else

cout<<num<<" is not a palindrome."<<endl;

}

**Output :**

****

* 1. **Write a program to implement a recursive function to calculate the sum of digits of a given number.**

#include <iostream>

using namespace std;

int sum(int num,int temp)

{

if(num==0)

return temp;

else

{

temp=temp+(num%10);

return sum(num/10,temp);

}

}

int main()

{

int num;

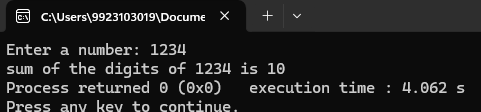
cout<<"Enter a number: ";

cin>>num;

cout<<"sum of the digits of "<<num<<" is "<<sum(num,0);

}

**Output :**

****

* 1. **Write a program to implement a recursive function to find the maximum and minimum elements in a given array.**

#include <iostream>

using namespace std;

int fmin(int arr[] ,int n)

{

if(n==1)

return arr[0];

return min(arr[n-1],fmin(arr,n-1));

}

int fmax(int arr[] ,int n)

{

if(n==1)

return arr[0];

return max(arr[n-1],fmin(arr,n-1));

}

int main()

{

int num;

cout<<"Input the size of array : ";

cin>>num;

int arr[num];

cout<"Input the elements : ";

for(int i=0;i<num;i++)

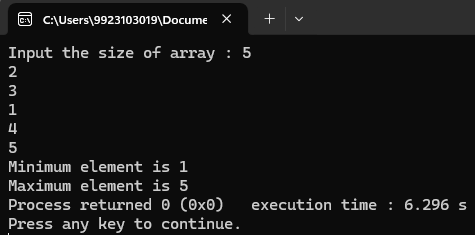
cin>>arr[i];

cout<<"Minimum element is "<<fmin(arr,num)<<endl;

cout<<"Maximum element is "<<fmax(arr,num);

}

**Output :**

****

* 1. **Write a program to reverse a string using recursion.**

#include <iostream>

using namespace std;

void reverseString(string &str,int start,int end)

{

if(start>=end)

return;

swap(str[start],str[end]);

reverseString(str,start+1,end-1);

}

int main()

{

string str;

cout<<"Input the string : ";

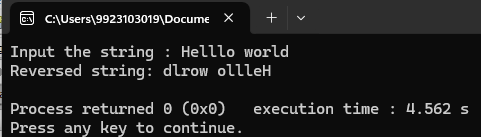
getline(cin,str);

reverseString(str,0,str.length()-1);

cout<<"Reversed string: "<<str<<endl;

}

**Outptut :**

****

* 1. **Write a program to implement a recursive function to reverse a linked list.**

#include <iostream>

using namespace std;

struct Node

{

int data;

Node\* next;

};

Node\* reverseList(Node\* head)

{

Node\* a=head;

Node\* b=head->next;

while(b->next!= NULL)

{

Node\* c=b->next;

b->next=a;

a=b;

b=c;

}

b->next=a;

head ->next= NULL;

head=b;

return b;

}

void push(Node\*\* head\_ref,int new\_data)

{

Node\* new\_node=new Node();

new\_node->data=new\_data;

new\_node->next=(\*head\_ref);

(\*head\_ref)=new\_node;

}

void printList(Node\* node)

{

while(node!=NULL)

{

cout<<node->data<<" ";

node=node->next;

}

cout<<endl;

}

int main()

{

Node\* head=NULL;

int n,a;

cout<<"Input the number of elements : ";

cin>>n;

cout<<"Input the elements : ";

for(int i=0;i<n;i++)

{

cin>>a;

push(&head,a);

}

cout<<"Original linked list: ";

printList(head);

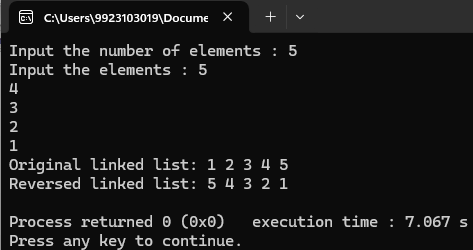
head=reverseList(head);

cout<<"Reversed linked list: ";

printList(head);

}

**Output :**

****

* 1. **Writeaprogramtoimplementarecursivefunctiontofindthegreatest commondivisor andLeastCommonMultiple.**

#include <iostream>

#include <string>

using namespace std;

int gcd(int a,int b)

{

if(a==0)

return b;

return gcd(b%a,a);

}

int lcm(int a,int b)

{

if(a%b==0)

return a;

else

return lcm(a+a,b);

}

int main()

{

int a,b;

cout<<"Input two numbers\na : ";

cin>>a;

cout<<"b : ";

cin>>b;

cout<<"GCD of "<<a<<" and "<<b<<" : "<<gcd(a,b);

cout<<"\nLCM of "<<a<<" and "<<b<<" : "<<lcm(a,b);

}

* 1. **Write a program to implement a recursive function to generate all permutations of a given set of numbers.**

#include <iostream>

using namespace std;

void printPermutation(int\* nums,int n)

{

for(int i=0; i<n; i++)

cout<<nums[i]<<" ";

cout<<endl;

}

void generatePermutations(int\* nums,int start,int n)

{

if(start>=n)

{

printPermutation(nums,n);

return;

}

for(int i=start; i<n; i++)

{

swap(nums[start],nums[i]);

generatePermutations(nums,start+1,n); swap(nums[start],nums[i]);

}

}

int main()

{

int n;

cout<<"Input the number of elements: ";

cin>>n;

int\* nums=new int[n];

cout<<"Input the elements : ";

for(int i=0; i<n; i++)

cin>>nums[i];

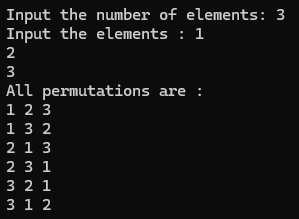
cout<<"All permutations are : \n";

generatePermutations(nums,0,n);

delete[] nums;

}

**Output :**

****

* 1. **Assume that you are given a string. You can now form continuous substrings fromthegivenstring.WriteaprogramtocountthenumberofUNIQUE continuoussetsofsubstringsthathavethesamestartingandending characters.(Youcanuseamixofrecursiveandnon-recursivefunctions).Eg:**

**Inputstring:“andisan”**

**Possiblesubstrings:**

**“a”,“an”,“and”,“andi”,“andis”,“andisa”,“andisan”,**

**“n”,“nd”,“ndi”,“ndis”,“ndisa”,“ndisan”,**

**“d”,“di”,“dis”,“disa”,“disan”,**

**“i”,“is”,“isa”,“isan”,**

**“s”,“sa”,“san”,**

**~~“a”,“an”,“n”,~~(alreadyconsideredearlier)**

**Output:7(“a”,“andisa”,“n”,“ndisan”,“d”,“i”,“s”)**

#include <iostream>

#include <string>

using namespace std;

bool isUnique(string substrings[],int count,const string &substr)

{

for(int i=0;i<count;++i)

if(substrings[i]==substr)

return false;

cout<<substr<<" ";

return true;

}

void findSubstrings(const string &str,string substrings[],int &count,int start,int end)

{

if(start>=end)

return;

if(str[start]==str[end-1])

{

string substr=str.substr(start,end-start);

if(isUnique(substrings,count,substr))

substrings[count++]=substr;

}

findSubstrings(str,substrings,count,start,end-1);

}

void countUniqueSubstrings(const string &str)

{

string substrings[100];

int count=0;

cout<<"Unique Substrings : ";

for(int i=0;i<str.size();++i)

findSubstrings(str,substrings,count,i,str.size());

cout<<"\nCount of Unique Substrings : "<<count;

}

int main()

{

cout<<"Input a string : ";

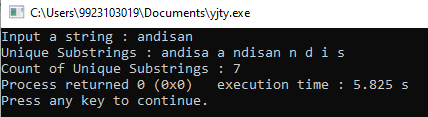
string s;

cin>>s;

countUniqueSubstrings(s);

}

**Output :**

****