**Sri Sri University, Cuttack, Odisha.**

**Faculty of Science**

|  |  |
| --- | --- |
| **Program: B.Sc. – Computer Science, Data Science, & Environmental Science**  **(2020-23 Batch)**  **Subject Code/Subject Name: Data Structure Laboratory**  **Assignment –XI** | |
| **Full Name of the Student:** | Vinayak Sanjay Chavan |
| **Full Roll Number:** | BCS - 011 |
| **Program:** | B.Sc. (Computer Sc.) / B.Sc. (Data Sc.) / B.Sc. (Env. Sc.) |
| **Date:** | 1st May, 2021 (10.00 AM – 12.00 Noon) |
| **Signature** |  |

|  |  |
| --- | --- |
| **All Questions are compulsory** | **Total Marks: 100** |

|  |  |
| --- | --- |
| **Question (s)** | **Maximum Marks** |
| 1. Implement the code for reversing a single linked list (SLL).   **Your code:** **#include<iostream>**  **#include <stdlib.h>**  **using namespace std;**  **typedef struct node {**  **int data;//element value**  **struct node \*link;//next node address for link**  **}NODE;**  **/\* Function to push nodes in a linked list. \*/ //add element after another**  **void push(NODE \*\*head, int element) {//using double pointer to store the data**  **NODE \*new\_node;**  **new\_node = (NODE\*)malloc(sizeof(NODE));//inner node storage alloacation**  **new\_node->data = element;//element value**  **new\_node->link = (\*head);//to link the node with another using address**  **(\*head) = new\_node;**  **}**    **/\* transverse the element from backside in a linked list. \*/**    **void reverse(NODE \*\*head) {**  **NODE \*next;//it is a variable that has no node linked**  **NODE\*previous = NULL;//another node that has no node**  **NODE \*current = (\*head);//if we have the head or first node already**  **while(current != NULL) {**  **next = current->link;//if current node is at 0 index then next will be at 1**  **current->link = previous;//then it will create the node at index -1 suppose (explanation)**  **previous = current;//now the element at current at index 0 will be stored at -1**  **current = next;//now the next element at index 1 will be act as current=1 and next will be at 2**  **}**  **(\*head) = previous;**  **}**  **/\* Function to print the nodes in a linked list. \*/**  **void display(NODE \*head) {**  **while(head != NULL) {**  **cout<<head->data<<" ";**  **head = head->link;**  **}**  **}**  **int main() {**  **NODE \*head = NULL;**  **push(&head, 1);**  **push(&head, 2);**  **push(&head, 3);**  **push(&head, 4);**  **push(&head, 5);**  **push(&head, 6);**  **cout << "List before reversing" << endl;**  **display(head);**  **reverse(&head);**  **cout << endl;**  **cout << "List after reversing"<<endl;**  **display(head);**  **return 0;**  **}**  **Screenshot of output:** | 20 |
| 1. One array of numbers to be sorted. The no of element of the array is a user input. Create the array dynamically, accept its members and sort the array using bubblesort algorithm. Also count the total number of swaps.   **Your code:** **/\* One array of numbers to be sorted. The no of element of the array**  **is a user input. Create the array dynamically, accept its members and**  **sort the array using bubblesort algorithm. Also count the total number of swaps. \*/**  **/\* One array of numbers to be sorted. The no of element of the array**  **is a user input. Create the array dynamically, accept its members and**  **sort the array using bubblesort algorithm. Also count the total number of swaps. \*/**  **#include <iostream>**  **#include <cstdlib>**  **using namespace std;**  **int linearsearch(int arr[], int n, int KEY, int size){**    **for(int i=0; i<n; i++){**  **if(arr[i] == KEY){**  **return i; //successfull**  **}**  **}**  **return -1; //NOT FOUND**  **}**  **void swap(int \*x, int \*y){**  **int temp;**    **temp = \*x;**  **\*x = \*y;**  **\*y = temp;**  **temp++;**  **}**  **void display(int arr[], int SIZE){**    **for(int i=0; i<SIZE; i++){**  **cout<<arr[i]<<" ";**  **}**  **}**  **void bubble\_sort(int arr[], int n){**  **int count = 0;**  **for(int i=0; i<n; i++){**  **int swapping = 0;**  **for(int j=0; j<n-i-1; j++){**  **if(arr[j] > arr[j+1]){**  **swap(&arr[j], &arr[j+1]);**  **count++;**  **swapping = 1;**  **}**  **}**  **if(!swapping)**  **break;**  **}**  **cout<<endl<<"how much time, swapping is done? ==>"<<count<<" "<<endl;**  **}**  **int main(){**  **int size;**  **cout<<"enter the size of the array: ";**  **cin>>size;**  **int\* ptr;**  **ptr = (int\*)malloc(sizeof(int)\*size);**  **cout<<"enter the array: ";**  **for(int i=0; i<size; i++){**  **cin>>ptr[i];**  **}**  **cout << "Array before Sorting: ";**  **display(ptr, size);**  **bubble\_sort(ptr, size);**  **cout << "Array after Sorting: ";**  **display(ptr, size);**  **}**  **Screenshot of output:** | 20 |
| 1. Implement selection sort algorithm. Call this function to sort an array of numbers created dynamically.   **Your code:** **/\* 3. Implement selection sort algorithm.**  **Call this function to sort an array of numbers created dynamically. \*/**  **#include <iostream>**  **#include <cstdlib>**  **using namespace std;**  **int linearsearch(int arr[], int n, int KEY, int size){**    **for(int i=0; i<n; i++){**  **if(arr[i] == KEY){**  **return i; //successfull**  **}**  **}**  **return -1; //NOT FOUND**  **}**  **void swap(int \*x, int \*y){**  **int temp;**    **temp = \*x;**  **\*x = \*y;**  **\*y = temp;**  **temp++;**  **}**  **void display(int arr[], int SIZE){**    **for(int i=0; i<SIZE; i++){**  **cout<<arr[i]<<" ";**  **}**  **}**  **void selection\_sort(int \*arr, int n){**  **for(int i=0; i<n-1; i++){**  **int k;**  **for(int j=k=i; j<n; j++){**  **if(arr[j] < arr[k]){**  **k = j;**  **}**  **}**  **swap(&arr[i], &arr[k]);**  **}**  **}**  **int main(){**  **int size;**  **cout<<"enter the size of the array: ";**  **cin>>size;**  **int\* ptr;**  **ptr = (int\*)malloc(sizeof(int)\*size);**  **cout<<"enter the array: ";**  **for(int i=0; i<size; i++){**  **cin>>ptr[i];**  **}**    **selection\_sort(ptr, size);**  **cout << "after Sorting: ";**  **display(ptr, size);**    **return 0;**  **}**  **Screenshot of output:** | 20 |
| 1. Write functions for sequential (linear) search and apply them to search an element in a sorted array. Implement both recursive and non-recursive code.   **Your code:** **//linear search**  **#include<iostream>**  **#include<cmath>**  **using namespace std;**  **int linear\_search(int arr[],int key,int size){**  **for(int i=0;i<size;i++){**  **if(arr[i]==key){**  **return i;**  **}**  **}**  **return -1;**  **}**  **int linear\_search\_rec(int arr[],int key,int size){**  **size=size-1;**  **if(size<0){**  **return -99;**  **}else if(arr[size]==key){**  **return size;**  **}else{**  **return linear\_search\_rec(arr,key,size);**  **}**  **}**  **int main(){**  **int size;**  **cout<<"Enter the size of the array: ";**  **cin>>size;**  **int arr[size];**  **cout<<"Enter the array: ";**  **for(int i=0;i<size;i++){**  **cin>>arr[i];**  **}**  **int search\_ele,index,index1;**  **cout<<"Enter the number to be searched: ";**  **cin>>search\_ele;**  **index=linear\_search(arr,search\_ele,size);**  **cout<<"The number is in: "<<index<<" index";**  **index1=linear\_search\_rec(arr,search\_ele,size);**  **cout<<"\nThe number is in: "<<index1<<" index";**  **return 0;**  **}**  **Screenshot of output:** |  |
| 1. Write functions for binary search and apply them to search an element in a sorted array. Implement both recursive and non-recursive code.   **Your code:**  **#include<iostream>**  **#include<cmath>**  **using namespace std;**  **int binary\_search\_rec(int arr[],int lower,int upper,int key){**  **int middle;**  **middle=(lower+upper)/2;**  **if(lower>upper){**  **return -1;**  **}else if(arr[middle]==key){**  **return middle;**  **}else if(key<arr[middle]){**  **return binary\_search\_rec(arr,lower,middle-1,key);**  **}else if(key>arr[middle]){**  **return binary\_search\_rec(arr,middle+1,upper,key);**  **}**  **}**  **int main(){**  **int size;**  **cout<<"Enter the size of the array: ";**  **cin>>size;**  **int arr[size];**  **cout<<"Enter the array: ";**  **for(int i=0;i<size;i++){**  **cin>>arr[i];**  **}**  **int search\_ele,index;**  **cout<<"Enter the number to be searched: ";**  **cin>>search\_ele;**  **index=binary\_search\_rec(arr,0,size-1,search\_ele);**  **cout<<"The number is in: "<<index;**  **return 0;**  **}**  **Screenshot of output:** | 20 |