Data Structure Lab Mannual

Subject Code: BCSL305

1. Design, Develop and Implement a menu-driven Program in C for the following Array operations
   1. Creating an Array of N Integer Elements
   2. Display of Array Elements with Suitable Headings
   3. Inserting an Element (ELEM) at a given valid Position (POS)
   4. Deleting an Element at a given valid Position(POS)
   5. Exit.

Support the program with functions for each of the above operations.

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

// Structure to represent a day

struct Day {

char \*name;

int date;

char \*activity;

};

// Function to create a day

struct Day createDay() {

struct Day day;

// Allocate memory for day name

day.name = (char \*)malloc(sizeof(char) \* 20);

printf("Enter day name: ");

scanf("%s", day.name);

printf("Enter date: ");

scanf("%d", &day.date);

// Allocate memory for activity description

day.activity = (char \*)malloc(sizeof(char) \* 100);

printf("Enter activity: ");

scanf(" %[^\n]s", day.activity);

return day;

}

// Function to read the calendar data from the keyboard

void readCalendar(struct Day \*calendar, int size) {

for (int i = 0; i < size; ++i) {

calendar[i] = createDay();

}

}

// Function to display the calendar

void displayCalendar(struct Day \*calendar, int size) {

for (int i = 0; i < size; ++i) {

printf("Day: %s\n", calendar[i].name);

printf("Date: %d\n", calendar[i].date);

printf("Activity: %s\n\n", calendar[i].activity);

}

}

// Function to free memory allocated for the calendar

void freeCalendar(struct Day \*calendar, int size) {

for (int i = 0; i < size; ++i) {

free(calendar[i].name);

free(calendar[i].activity);

}

}

int main() {

int size;

// Get the number of days in a week

printf("Enter the number of days in a week: ");

scanf("%d", &size);

// Dynamically allocate memory for the calendar

struct Day \*calendar = (struct Day \*)malloc(sizeof(struct Day) \* size);

// Check if memory allocation is successful

if (calendar == NULL) {

printf("Memory allocation failed.\n");

return 1; // Exit the program with an error code

}

// Read calendar data from the keyboard

readCalendar(calendar, size);

// Display calendar details

printf("\nCalendar Details:\n");

displayCalendar(calendar, size);

// Free memory allocated for the calendar

freeCalendar(calendar, size);

free(calendar);

return 0; // Exit the program successfully

}

Program 2: Develop a Program in C for the following operations on strings:

a.Read a main string(STR), a Pattern String(PAT) and Replace String (REP) .

b- Perform Pattern matching Operation: find and replace all occurances of PAT does not exist in STR.

support the program with functions for each of the above operations. Dont use built in functions.

#include <stdio.h>

#include <string.h>

// Function to read a string

void readString(char \*str, const char \*prompt) {

printf("%s", prompt);

scanf("%s", str);

}

// Function to perform pattern matching and replacement

void patternMatching(char \*mainStr, const char \*pattern, const char \*replace) {

char result[1000]; // Assuming a fixed size for the result, adjust as needed

int mainLen = strlen(mainStr);

int patLen = strlen(pattern);

int repLen = strlen(replace);

int i, j, k;

for (i = 0; i <= mainLen - patLen; ) {

j = 0;

// Check for pattern match

while (j < patLen && mainStr[i + j] == pattern[j]) {

j++;

}

// If pattern found, replace it and move index accordingly

if (j == patLen) {

for (k = 0; k < repLen; k++) {

result[i + k] = replace[k];

}

i += repLen;

} else {

result[i] = mainStr[i];

i++;

}

}

// Copy remaining characters from mainStr to result

while (i < mainLen) {

result[i] = mainStr[i];

i++;

}

// Null-terminate the result

result[i] = '\0';

// Copy result back to mainStr

strcpy(mainStr, result);

}

int main() {

char mainStr[1000], pattern[100], replace[100];

// Read main string

readString(mainStr, "Enter the main string: ");

// Read pattern string

readString(pattern, "Enter the pattern string: ");

// Read replace string

readString(replace, "Enter the replace string: ");

// Perform pattern matching and replacement

patternMatching(mainStr, pattern, replace);

// Display the result

printf("Result after pattern matching and replacement: %s\n", mainStr);

return 0;

}

3. Develop a menu driven program in C for the following operations on STACK of integers (Array Implementation of stack with Maximum size MAX)

a. Push an element on to stack

b. Pop an element from stack

c. Demonstrate how stack can be used to check Palindrome

d.Demonstrate Overflow and Underflow situations on stack

e.Display the status of stack

f. exit.

Support the program with appropriate functions for each of the above operation

#include<stdio.h>

#include<stdlib.h>

#define MAX 5

int s[MAX];

int top = -1;

void push(int item);

int pop();

void palindrome();

void display();

void main()

{

            int choice, item;

            while(1)

            {

                        printf("\n\n\n\n~~~~~~Menu~~~~~~ : ");

                        printf("\n=>1.Push an Element to Stack and Overflow demo ");

                        printf("\n=>2.Pop an Element from Stack and Underflow demo");

                        printf("\n=>3.Palindrome demo ");

                        printf("\n=>4.Display ");

                        printf("\n=>5.Exit");

                        printf("\nEnter your choice: ");

                        scanf("%d", &choice);

                        switch(choice)

                        {

                                    case 1:             printf("\nEnter an element to be pushed: ");

                                                            scanf("%d", &item);

**push(item);**

                                                            break;

                                    case 2:             **item = pop();**

                                                            if(item != -1)

                                                                        printf("\nElement popped is: %d", item);

                                                            break;

                                    case 3:             **palindrome();**

                                                            break;

                                    case 4:             **display();**

                                                            break;

                                    case 5:             exit(1);

                                    default:            printf("\nPlease enter valid choice ") ;

                                                            break;

                    }

            }

}

void push(int item)

{

            if(top == MAX-1)

            {

                        printf("\n~~~~Stack overflow~~~~");

                        return;

            }

            top = top + 1 ;

            s[top] = item;

}

int pop()

{

            int item;

            if(top == -1)

            {

                        printf("\n~~~~Stack underflow~~~~");

                        return -1;

            }

            item = s[top];

            top = top - 1;

            return item;

}

void display()

{

            int i;

            if(top == -1)

            {

                        printf("\n~~~~Stack is empty~~~~");

                        return;

            }

            printf("\nStack elements are:\n ");

            for(i=top; i>=0 ; i--)

                        printf("| %d |\n", s[i]);

}

void palindrome()

{

            int flag=1,i;

            printf("\nStack content are:\n");

            for(i=top; i>=0 ; i--)

                        printf("| %d |\n", s[i]);

            printf("\nReverse of stack content are:\n");

            for(i=0; i<=top; i++)

                        printf("| %d |\n", s[i]);

            for(i=0; i<=top/2; i++)

            {

                        if( s[i] != s[top-i] )

                        {

                                    flag = 0;

                                    break;

                        }

            }

            if(flag == 1)

            {

                        printf("\nIt is palindrome number");

            }

            else

            {

                        printf("\nIt is not a palindrome number");

            }

}

Program 4: Develop a program in C for converting an Infix expression to Postfix expression. Program shouls support for both parenthesezed expressions with the operators +,-,\*,/, ^ and alphanumeric operands.

#include<stdio.h>

#include<ctype.h>

char stack[100];

int top = -1;

void push(char x)

{

stack[++top] = x;

}

char pop()

{

if(top == -1)

return -1;

else

return stack[top--];

}

int priority(char x)

{

if(x == '(')

return 0;

if(x == '+' || x == '-')

return 1;

if(x == '\*' || x == '/')

return 2;

return 0;

}

int main()

{

char exp[100];

char \*e, x;

printf("Enter the expression : ");

scanf("%s",exp);

printf("\n");

e = exp;

while(\*e != '\0')

{

if(isalnum(\*e))

printf("%c ",\*e);

else if(\*e == '(')

push(\*e);

else if(\*e == ')')

{

while((x = pop()) != '(')

printf("%c ", x);

}

else

{

while(priority(stack[top]) >= priority(\*e))

printf("%c ",pop());

push(\*e);

}

e++;

}

while(top != -1)

{

printf("%c ",pop());

}return 0;

}

1. Develop a c program for the following stack applications a) evaluation of suffix expression with single digit operands and operators +,-,\*,/,%,^ b)solving tower honoi problem with n disks

#include<stdio.h>

#include<math.h>

#include<string.h>

#include<ctype.h>

double compute(char symbol,double op1,double op2);

void main()

{

double s[20],res,op1,op2;

int top=-1,i;

char postfix[20],symbol;

printf("Enter the Postfix Expression\n");

scanf("%s",postfix);

for(i=0;i<strlen(postfix);i++)

{

symbol=postfix[i];

if(isdigit(symbol))

s[++top]=symbol-'0';

else

{

op2=s[top--];

op1=s[top--];

res=compute(symbol,op1,op2);

s[++top]=res;

}

}

res=s[top--];

printf("The result %f \n",res);

}

double compute(char symbol,double op1,double op2)

{

switch(symbol)

{

case '+':return op1+op2;

case '-':return op1-op2;

case '/':return op1/op2;

case '\*':return op1\*op2;

/\*case '^':

case '$':return (pow(op1,op2)) ;\*/

}

}

B .#include<stdio.h>

#include<stdlib.h>

void TOH( int n, char A, char B, char C);

void main()

{

int n;

printf("Enter number of rings:\n");

scanf("%d", &n);

TOH(n, 'A', 'C', 'B');

}

void TOH( int n, char A, char B, char C)

{

if(n==1)

printf("Move from %c to %c\n",A, B);

else

{

TOH(n-1, A, C, B);

TOH(1, A, B, C);

TOH(n-1, C, B, A);

}

}

Output

Enter number of rings:

3

Move from A to C

Move from A to B

Move from C to B

Move from A to C

Move from B to A

Move from B to C

Move from A to C

6. Develop a menu driven program in C for the following operations on circular QUEUE of characters (Array implementation of Queue with maximum size MAX)

a. Insert an element on circular queue.

b. Delete an element from circular queue.

c. Demonstrate Overflow and Underflow situations on circular queue.

d. Display the status of circular queue.

e. Exit

#include <stdio.h>

#include <stdlib.h>

#define max 5

int q[max],f=-1,r=-1;

void ins()

{

if(f==(r+1)%max)

printf("\nQueue overflow");

else

{

if(f==-1)

f++;

r=(r+1)%max;

printf("\nEnter element to be inserted:");

scanf("%d",&q[r]);

}

}

void del()

{

if(r==-1)

printf("\nQueue underflow");

else

{

printf("\nElemnt deleted is:%d",q[f]);

if(f==r)

f=r=-1;

else

f=(f+1)%max;

}

}

void disp()

{

if(f==-1)

printf("\nQueue empty");

else

{

int i;

printf("\nQueue elements are:\n");

for(i=f;i!=r;i=(i+1)%max)

printf("%d\t",q[i]);

printf("%d",q[i]);

printf("\nFront is at:%d\nRear is at:%d",q[f],q[r]);

}

}

int main()

{

printf("\nCircular Queue operations");

printf("\n1.Insert");

printf("\n2.Delete");

printf("\n3.Display");

printf("\n4.Exit");

int ch;

do{

printf("\nEnter choice:");

scanf("%d",&ch);

switch(ch)

{

case 1:ins();break;

case 2:del();break;

case 3:disp();break;

case 4:exit(0);

default:printf("\nInvalid choice...!");

}

}while(1);

return 0;

}

7. Design, Develop and Implement a menu driven Program in C for the following operations on Singly Linked List (SLL) of Student Data with the fields: USN, Name, Branch, Sem, PhNo

* 1. Create a SLL of N Students Data by using front insertion.
  2. Display the status of SLL and count the number of nodes in it
  3. Perform Insertion / Deletion at End of SLL
  4. Perform Insertion / Deletion at Front of SLL(Demonstration of stack)
  5. Exit.

#include<string.h>

#include<stdio.h>

#include<stdlib.h>

struct stud

{

char usn[11],name[15],branch[4],phno[11];

int sem;

struct stud \*next;

}\*f=NULL,\*r=NULL,\*t=NULL;

void ins(int ch)

{

t=(struct stud\*)malloc(sizeof(struct stud));

printf("\nEnter USN:");

scanf("%s",t->usn);

printf("Enter Name:");

scanf("%s",t->name);

printf("Enter Branch:");

scanf("%s",t->branch);

printf("Enter Sem:");

scanf("%d",&t->sem);

printf("Enter Phno:");

scanf("%s",t->phno);

t->next=NULL;

if(!r)

f=r=t;

else

{

if(ch)

{

r->next=t;

r=t;

}

else

{

t->next=f;

f=t;

}

}

}

void del(int ch)

{

if(!f)

printf("\nList Empty");

else

{

struct stud \*t1;

if(f==r)

{

t1=f;

f=r=NULL;

}

else if(ch)

{

t1=r;

for(t=f;t->next!=r;t=t->next)

r=t;

r->next=NULL;

}

else

{

t1=f;

f=f->next;

}

printf("\nElement deleted is:\n");

printf("USN:%s\nName:%s\nBranch:%s\nSem:%d\nPhno:%s\n",t1->usn,t1->name,t1->branch,t1->sem,t1->phno);

free(t1);

}

}

void disp()

{

if(!f)

printf("\nList Empty!!!");

else

printf("\nList elements are:\n");

for(t=f;t;t=t->next)

printf("\nUSN:%s\nName:%s\nBranch:%s\nSem:%d\nPhno:%s\n",t->usn,t->name,t->branch,t->sem,t->phno);

}

void main()

{

int ch,n,i;

printf("\n........Menu..........,\n");

printf("1.Create\n");

printf("2.Display\n");

printf("3.Insert at end\n");

printf("4.Delete at end\n");

printf("5.Insert at beg\n");

printf("6.Delete at beg\n");

printf("7.Exit\n");

while(1)

{

printf("\nEnter choice:");

scanf("%d",&ch);

switch(ch)

{

case 1: printf("\nEnter no. of nodes:");

scanf("%d",&n);

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

ins(0);

break;

case 2:disp();break;

case 3:ins(1);break;

case 4:del(1);break;

case 5:ins(0);break;

case 6:del(0);break;

case 7:exit(0);

default:printf("\nInvalid choice!!!!");

}

}

}

8. Design, Develop and Implement a menu driven Program in C for the following operations on Doubly Linked List (DLL) of Employee Data with the fields: SSN, Name, Dept, Designation, Sal, PhNo

* 1. Create a DLL of N Employees Data by using end insertion.
  2. Display the status of DLL and count the number of nodes in it
  3. Perform Insertion and Deletion at End of DLL
  4. Perform Insertion and Deletion at Front of DLL
  5. Demonstrate how this DLL can be used as Double Ended Queue
  6. Exit

#include<string.h>

int count=0;

struct node

{

struct node \*prev;

int ssn,phno;

float sal;

char name[20],dept[10],desg[20];

struct node \*next;

}\*h,\*temp,\*temp1,\*temp2,\*temp4;

void create()

{

int ssn,phno;

float sal;

char name[20],dept[10],desg[20];

temp =(struct node \*)malloc(sizeof(struct node));

temp->prev = NULL;

temp->next = NULL;

printf("\n Enter ssn,name,department, designation, salary and phno of employee : ");

scanf("%d %s %s %s %f %d", &ssn, name,dept,desg,&sal, &phno);

temp->ssn = ssn;

strcpy(temp->name,name);

strcpy(temp->dept,dept);

strcpy(temp->desg,desg);

temp->sal = sal;

temp->phno = phno;

count++;

}

void insertbeg()

{

if (h == NULL)

{

create();

h = temp;

temp1 = h;

}

else

{

create();

temp->next = h;

h->prev = temp;

h = temp;

}

}

void insertend()

{

if(h==NULL)

{

create();

h = temp;

temp1 = h;

}

else

{

create();

temp1->next = temp;

temp->prev = temp1;

temp1 = temp;

}

}

void displaybeg()

{

temp2 =h;

if(temp2 == NULL)

{

printf("List empty to display \n");

return;

}

printf("\n Linked list elements from begining : \n");

while (temp2!= NULL)

{

printf("%d %s %s %s %f %d\n", temp2->ssn, temp2->name,temp2->dept,

temp2->desg,temp2->sal, temp2->phno );

temp2 = temp2->next;

}

printf(" No of employees = %d ", count);

}

int deleteend()

{

struct node \*temp;

temp=h;

if(temp->next==NULL)

{

free(temp);

h=NULL;

return 0;

}

else

{

temp2=temp1->prev;

temp2->next=NULL;

printf("%d %s %s %s %f %d\n", temp1->ssn, temp1->name,temp1->dept,

temp1->desg,temp1->sal, temp1->phno );

free(temp1);

}

count--;

return 0;

}

int deletebeg()

{

struct node \*temp;

temp=h;

if(temp->next==NULL)

{

free(temp);

h=NULL;

}

else

{

h=h->next;

printf("%d %s %s %s %f %d", temp->ssn, temp->name,temp->dept,

temp->desg,temp->sal, temp->phno );

free(temp);

}

count--;

return 0;

}

void main()

{

int ch,n,i;

h=NULL;

temp = temp1 = NULL;

printf("-----------------MENU--------------------\n");

printf("\n 1 - create a DLL of n emp");

printf("\n 2 - Display from beginning");

printf("\n 3 - Insert at end");

printf("\n 4 - delete at end");

printf("\n 5 - Insert at beg");

printf("\n 6 - delete at beg");

printf("\n 7 - exit\n");

printf("------------------------------------------\n");

while (1)

{

printf("\n Enter choice : ");

scanf("%d", &ch);

switch (ch)

{

case 1:

printf("\n Enter no of employees : ");

scanf("%d", &n);

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

insertend();

break;

case 2:

displaybeg();

break;

case 3:

insertend();

break;

case 4:

deleteend();

break;

case 5:

insertbeg();

break;

case 6:

deletebeg();

break;

case 7:

exit(0);

default:

printf("wrong choice\n");

}

}

}

9. Design, Develop and Implement a Program in C for the following operations on Singly Circular Linked List (SCLL) with header nodes

1.Represent and Evaluate a Polynomial P(x,y,z) = 6x2y2z-4yz5+3x3yz+2xy5z-2xyz3

2. Find the sum of two polynomials POLY1(x,y,z) and POLY2(x,y,z) and store the result in POLYSUM(x,y,z)

Support the program with appropriate functions for each of the above operations.

#include<stdio.h>

#include<math.h>

#include<stdlib.h>

#include<math.h>

typedef struct node

{

int expo,coef;

struct node \*next;

}node;

/\*FUNCTION PROTOTYPE\*/

node \* insert(node \*,int,int);

node \* create();

node \* add(node \*p1,node \*p2);

int eval(node \*p1);

void display(node \*head);

node \*insert(node\*head,int expo1,int coef1)

{

node \*p,\*q;

p=(node \*)malloc(sizeof(node));

p->expo=expo1;

p->coef=coef1;

p->next=NULL;

if(head==NULL)

{

head=p;

head->next=head;

return(head);

}

if(expo1>head->expo)

{

p->next=head->next;

head->next=p;

head=p;

return(head);

}

if(expo1==head->expo)

{

head->coef=head->coef+coef1;

return(head);

}

q=head;

while(q->next!=head&&expo1>=q->next->expo)

q=q->next;

if(p->expo==q->expo)

q->coef=q->coef+coef1;

else

{

p->next=q->next;

q->next=p;

}

return(head);

}

node \*create()

{

int n,i,expo1,coef1;

node \*head=NULL;

printf("\n\nEnter no of terms of polynomial==>");

scanf("%d",&n);

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

{

printf("\n\nEnter coef & expo==>");

scanf("%d%d",&coef1,&expo1);

head=insert(head,expo1,coef1);

}

return(head);

}

node \*add(node \*p1,node \*p2)

{

node \*p;

node \*head=NULL;

printf("\n\n\nAddition of polynomial==>");

p=p1->next;

do

{

head=insert(head,p->expo,p->coef);

p=p->next;

}while(p!=p1->next);

p=p2->next;

do

{

head=insert(head,p->expo,p->coef);

p=p->next;

}while(p!=p2->next);

return(head);

}

int eval(node \*head)

{

node \*p;

int x,ans=0;

printf("\n\nEnter the value of x=");

scanf("%d",&x);

p=head->next;

do

{

ans=ans+p->coef\*pow(x,p->expo);

p=p->next;

}while(p!=head->next);

return(ans);

}

void display(node \*head)

{

node \*p,\*q;

int n=0;

q=head->next;

p=head->next;

do

{

n++;

q=q->next;

}while(q!=head->next);

printf("\n\n\tThe polynomial is==>");

do

{

if(n-1)

{

printf("%dx^(%d) + ",p->coef,p->expo);

p=p->next;

}

else

{

printf(" %dx^(%d)",p->coef,p->expo);

p=p->next;

}

n--;

} while(p!=head->next);

}

void main()

{

int a,x,ch;

node \*p1,\*p2,\*p3;

p1=p2=p3=NULL;

while(1)

{

printf("\n\t----------------<< MENU >>---------------");

printf("\n\tPolynomial Operations :");

printf(" 1.Add");

printf("\n\t\t\t\t2.Evaluate");

printf("\n\t\t\t\t3.Exit");

printf("\n\t------------------------------------------- ");

printf("\n\n\n\tEnter your choice==>");

scanf("%d",&ch);

switch(ch)

{

case 1 :

p1=create();

display(p1);

p2=create();

display(p2);

p3=add(p1,p2);

display(p3);

break;

case 2 :

p1=create();

display(p1);

a=eval(p1);

printf("\n\nValue of polynomial=%d",a);

break;

case 3 :

exit(0);

break;

default :

printf("\n\n\t invalid choice");

break;

}

}

}

10. Design, Develop and Implement a menu driven Program in C for the following operations on Binary Search Tree(BST) of Integers

a. Create a BST of N Integers: 6, 9, 5, 2, 8, 15, 24, 14, 7, 8, 5, 2

b. Traverse the BST in Inorder, Preorder and Post Order

c. Search the BST for a given element (KEY) and report the appropriate message

d.Exit.

#include <stdio.h>

#include <stdlib.h>

int flag=0;

typedef struct BST

{

int data;

struct BST \*lchild,\*rchild;

} node;

/\*FUNCTION PROTOTYPE\*/

void insert(node \*, node \*);

void inorder(node \*);

void preorder(node \*);

void postorder(node \*);

node \*search(node \*, int, node \*\*);

void main()

{

int choice;

int ans =1;

int key;

node \*new\_node, \*root, \*tmp, \*parent;

node \*get\_node();

root = NULL;

printf("\nProgram For Binary Search Tree ");

do

{

printf("\n1.Create");

printf("\n2.Search");

printf("\n3.Recursive Traversals");

printf("\n4.Exit");

printf("\nEnter your choice :");

scanf("%d", &choice);

switch (choice)

{

case 1:

do

{

new\_node = get\_node();

printf("\nEnter The Element ");

scanf("%d", &new\_node->data);

if (root == NULL) /\* Tree is not Created \*/

root = new\_node;

else

insert(root, new\_node);

printf("\nWant To enter More Elements?(1/0)");

scanf("%d",&ans);

} while (ans);

break;

case 2:

printf("\nEnter Element to be searched :");

scanf("%d", &key);

tmp = search(root, key, &parent);

if(flag==1)

{

printf("\nParent of node %d is %d", tmp->data, parent->data);

}

else

{

printf("\n The %d Element is not Present",key);

}

flag=0;

break;

case 3:

if (root == NULL)

printf("Tree Is Not Created");

else

{

printf("\nThe Inorder display :");

inorder(root);

printf("\nThe Preorder display : ");

preorder(root);

printf("\nThe Postorder display : ");

postorder(root);

}

break;

}

}

while (choice != 4);

}

/\*Get new Node \*/

node \*get\_node()

{

node \*temp;

temp = (node \*) malloc(sizeof(node));

temp->lchild = NULL;

temp->rchild = NULL;

return temp;

}

/\*This function is for creating a binary search tree \*/

void insert(node \*root, node \*new\_node)

{

if (new\_node->data < root->data)

{

if(root->lchild==NULL)

root->lchild=new\_node;

else

insert(root->lchild, new\_node);

}

if (new\_node->data > root->data)

{

if (root->rchild == NULL)

root->rchild = new\_node;

else

insert(root->rchild, new\_node);

}

}

/\*This function is for searching the node from binary Search Tree\*/

node \*search(node \*root, int key, node \*\*parent)

{

node \*temp;

temp = root;

while (temp != NULL)

{

if (temp->data == key)

{

printf("\nThe %d Element is Present", temp->data);

flag=1;

return temp;

}

\*parent = temp;

if (temp->data > key)

temp = temp->lchild;

else

temp = temp->rchild;

}

return NULL;

}

/\*This function displays the tree in inorder fashion \*/

void inorder(node \*temp)

{

if (temp != NULL)

{

inorder(temp->lchild);

printf("%d\t", temp->data);

inorder(temp->rchild);

}

}

/\*This function displays the tree in preorder fashion \*/

void preorder(node \*temp)

{

if (temp != NULL)

{

printf("%d\t", temp->data);

preorder(temp->lchild);

preorder(temp->rchild);

}

}

/\*This function displays the tree in postorder fashion \*/

void postorder(node \*temp)

{

if (temp != NULL)

{

postorder(temp->lchild);

postorder(temp->rchild);

printf("%d\t", temp->data);

}

}

11.Design, Develop and Implement a Program in C for the following operations on Graph(G) of Cities

* 1. Create a Graph of N cities using Adjacency Matrix.
  2. Print all the nodes reachable from a given starting node in a digraph using DFS/BFS method

#include <stdio.h>

#include <stdlib.h>

int a[20][20],q[20],visited[20],reach[10],n,i,j,f=0,r= -1,count=0;

void bfs(int v)

{

for(i=1;i<=n;i++)

if(a[v][i] && !visited[i])

q[++r]=i;

if(f<=r)

{

visited[q[f]]=1;

bfs(q[f++]);

}

}

void dfs(int v)

{

int i;

reach[v]=1;

for(i=1;i<=n;i++)

{

if(a[v][i] && !reach[i])

{

printf("\n %d->%d",v,i);

count++;

dfs(i);

}

}

}

void main()

{

int v, choice;

printf("\n Enter the number of vertices:");

scanf("%d",&n);

for(i=1;i<=n;i++)

{

q[i]=0;

visited[i]=0;

}

for(i=1;i<=n-1;i++)

reach[i]=0;

printf("\n Enter graph data in matrix form:\n");

for(i=1;i<=n;i++)

for(j=1;j<=n;j++)

scanf("%d",&a[i][j]);

printf("1.BFS\n 2.DFS\n 3.Exit\n");

scanf("%d",&choice);

switch(choice)

{

case 1:

printf("\n Enter the starting vertex:");

scanf("%d",&v);

bfs(v);

if((v<1)||(v>n))

{

printf("\n Bfs is not possible");

}

else

{

printf("\n The nodes which are reachable from %d:\n",v);

for(i=1;i<=n;i++)

if(visited[i])

printf("%d\t",i);

}

break;

case 2:

dfs(1);

if(count==n-1)

printf("\n Graph is connected");

else

printf("\n Graph is not connected");

break;

case 3:

exit(0);

}

}

1. Given a File of N employee records with a set K of Keys(4-digit) which uniquely determine the records in file F. Assume that file F is maintained in memory by a Hash Table(HT) of m memory locations with L as the set of memory addresses (2-digit) of locations in HT.

Let the keys in K and addresses in L are Integers. Design and develop a program in C that uses Hash function H: K → L as H(K)=K mod m (remainder method), and implement hashing technique to map a given key K to the address space L. Resolve the collision  
(if any) using linear probing.

#include <stdio.h>

#include <stdlib.h>

#define MAX 100

/\*FUNCTION PROTOTYPE \*/

int create(int);

void linear\_prob(int[], int, int);

void display (int[]);

void main()

{

int a[MAX],num,key,i;

int ans=1;

printf(" collision handling by linear probing : \n");

for (i=0;i<MAX;i++)

{

a[i] = -1;

}

do

{

printf("\n Enter the data");

scanf("%4d", &num);

key=create(num);

linear\_prob(a,key,num);

printf("\n Do you wish to continue ? (1/0) ");

scanf("%d",&ans);

}while(ans);

display(a);

}

int create(int num)

{

int key;

key=num%100;

return key;

}

void linear\_prob(int a[MAX], int key, int num)

{

int flag, i, count=0;

flag=0;

if(a[key]== -1)

{

a[key] = num;

}

else

{

printf("\nCollision Detected...!!!\n");

i=0;

while(i<MAX)

{

if (a[i]!=-1)

count++;

i++;

}

printf("Collision avoided successfully using LINEAR PROBING\n");

if(count == MAX)

{

printf("\n Hash table is full");

display(a);

exit(1);

}

for(i=key+1; i<MAX; i++)

if(a[i] == -1)

{

a[i] = num;

flag =1;

break;

}

//for(i=0;i<key;i++)

i=0;

while((i<key) && (flag==0))

{

if(a[i] == -1)

{

a[i] = num;

flag=1;

break;

}

i++;

}

}

}

void display(int a[MAX])

{

int i,choice;

printf("1.Display ALL\n 2.Filtered Display\n");

scanf("%d",&choice);

if(choice==1)

{

printf("\n the hash table is\n");

for(i=0; i<MAX; i++)

printf("\n %d %d ", i, a[i]);

}

else

{

printf("\n the hash table is\n");

for(i=0; i<MAX; i++)

if(a[i]!=-1)

{

printf("\n %d %d ", i, a[i]);

continue;

}

}

}