**NEW HORIZON COLLEGE OF ENGINEERING BANGALORE**

**Department of Information Science and Engineering**

**DATA STRUCTURES USING C LABORATORY**

**19ISL38**

**2019-20 SEMESTER-III**

1. **Design, Develop and Implement a menu driven Program in C for the following Array operations  
   a. Creating an Array of N Integer Elements  
   b. Display of Array Elements with Suitable Headings  
   c. Inserting an Element (ELEM) at a given valid Position (POS)   
   d. Deleting an Element at a given valid Position(POS)  
   e. Exit.   
   Support the program with functions for each of the above operations.**

#include<stdio.h>

#include<stdlib.h>

#define MAX 5

int a[MAX], pos, elem;

int n = 0;

void create();

void display();

void insert();

void delete();

void main()

{

            int choice;

            while(1)

            {

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

                        printf("\n=>1. Create an array of N integers");

                        printf("\n=>2. Display of array elements");

                        printf("\n=>3. Insert ELEM at a given POS");

                         printf("\n=>4. Delete an element at a given POS");

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

                         printf("\nEnter your choice: ");

                        scanf("%d", &choice);

                        switch(choice)

                        {

                                    case 1:             create();

                                                            break;

                                    case 2:             display();

                                                            break;

                                    case 3:             insert();

                                                            break;

                                    case 4:             delete();

                                                            break;

                                    case 5:             exit(1);

                                                            break;

                                    default:            printf("\n Please enter a valid choice:");

                        }

            }

}

void create()

{

            int i;

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

            scanf("%d", &n);

            printf("\n Enter the elements: ");

            for(i=0; i<n; i++)

            {

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

            }

}

void display()

{

            int i;

            if(n = = 0)

            {

                        printf("\n No elements to display");

                        return;

            }

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

            for(i=0; i<n; i++)

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

}

void insert()

{

            int i;

            if(n = = MAX)

            {

                        printf("\n Array is full. Insertion is not possible");

                        return;

            }

            do

            {

                        printf("\n Enter a valid position where element to be inserted:    ");

                        scanf("%d", &pos);

            }while(pos > n);

            printf("\n Enter the value to be inserted:   ");

            scanf("%d", &elem);

            for (i=n-1; i>=pos ; i--)

            {

                        a[i+1] = a[i];

            }

a[pos] = elem;

            n = n+1;

            display();

}

void delete()

{

            int i;

            if(n == 0)

            {

                        printf("\n Array is empty and no elements to delete");

                        return;

            }

            do

            {

                        printf("\n Enter a valid position from where element to be deleted:    ");

                        scanf("%d", &pos);

            }while(pos>=n);

**elem = a[pos];**

            printf("\n Deleted element is : %d \n", elem);

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

            {

                        a[i] = a[i+1];

            }

**n = n-1;**

            display();

}

***Output:***

~~~~MENU~~~~

=>1. Create an array of N integers

=>2. Display of array elements

=>3. Insert ELEM at a given POS

=>4. Delete an element at a given POS

=>5. Exit

Enter your choice: **1**

**Enter the number of elements: 3**

**Enter the elements: 10 20 30**

~~~~MENU~~~~

=>1. Create an array of N integers

=>2. Display of array elements

=>3. Insert ELEM at a given POS

=>4. Delete an element at a given POS

=>5. Exit

Enter your choice: **2**

**Array elements are: 10   20      30**

~~~~MENU~~~~

=>1. Create an array of N integers

=>2. Display of array elements

=>3. Insert ELEM at a given POS

=>4. Delete an element at a given POS

=>5. Exit

Enter your choice: **3**

**Enter a valid position where element to be inserted:    5**

**Enter a valid position where element to be inserted:    4**

**Enter a valid position where element to be inserted:    3**

**Enter the value to be inserted:   40**

**Array elements are: 10   20      30      40**

~~~~MENU~~~~

=>1. Create an array of N integers

=>2. Display of array elements

=>3. Insert ELEM at a given POS

=>4. Delete an element at a given POS

=>5. Exit

Enter your choice: **3**

**Enter a valid position where element to be inserted:    4**

**Enter the value to be inserted:   50**

**Array elements are: 10   20      30      40      50**

~~~~MENU~~~~

=>1. Create an array of N integers

=>2. Display of array elements

=>3. Insert ELEM at a given POS

=>4. Delete an element at a given POS

=>5. Exit

Enter your choice: **3**

**Array is full. Insertion is not possible**

~~~~MENU~~~~

=>1. Create an array of N integers

=>2. Display of array elements

=>3. Insert ELEM at a given POS

=>4. Delete an element at a given POS

=>5. Exit

Enter your choice: **4**

**Enter a valid position from where element to be deleted:    5**

**Enter a valid position from where element to be deleted:    6**

**Enter a valid position from where element to be deleted:    4**

**Deleted element is: 50**

**Array elements are: 10   20      30      40**

~~~~MENU~~~~

=>1. Create an array of N integers

=>2. Display of array elements

=>3. Insert ELEM at a given POS

=>4. Delete an element at a given POS

=>5. Exit

Enter your choice: **4**

**Enter a valid position from where element to be deleted:    2**

**Deleted element is: 30**

**Array elements are: 10   20      40**

~~~~MENU~~~~

=>1. Create an array of N integers

=>2. Display of array elements

=>3. Insert ELEM at a given POS

=>4. Delete an element at a given POS

=>5. Exit

Enter your choice: **4**

**Enter a valid position from where element to be deleted:    1**

**Deleted element is: 20**

**Array elements are: 10   40**

~~~~MENU~~~~

=>1. Create an array of N integers

=>2. Display of array elements

=>3. Insert ELEM at a given POS

=>4. Delete an element at a given POS

=>5. Exit

Enter your choice: **4**

**Enter a valid position from where element to be deleted:    0**

**Deleted element is: 10**

**Array elements are: 40**

~~~~MENU~~~~

=>1. Create an array of N integers

=>2. Display of array elements

=>3. Insert ELEM at a given POS

=>4. Delete an element at a given POS

=>5. Exit

Enter your choice: **4**

**Enter a valid position from where element to be deleted:    0**

**Deleted element is: 40**

**No elements to display**

~~~~MENU~~~~

=>1. Create an array of N integers

=>2. Display of array elements

=>3. Insert ELEM at a given POS

=>4. Delete an element at a given POS

=>5. Exit

Enter your choice: **4**

**Array is empty and no elements to delete**

~~~~MENU~~~~

=>1. Create an array of N integers

=>2. Display of array elements

=>3. Insert ELEM at a given POS

=>4. Delete an element at a given POS

=>5. Exit

Enter your choice: **5**

**2**. **Design, Develop and Implement a Program in C for the following operations on Strings.**

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

**b. Perform Pattern Matching Operation: Find and Replace all occurrences of PAT in STR with REP if PAT exists in STR. Report suitable messages in case PAT does not exist in STR.**

**Support the program with functions for each of the above operations. Don't use Built-in functions.**

#include<stdio.h>

char str[50], pat[20], rep[20], ans[50];

int     c=0, m=0, i=0, j=0, k, flag=0;

void stringmatch()

{

                 while(str[c] !='\0')

                {

                                if(str[m] == pat[i])

                                {

                                                i++;

                                                m++;

                                                if(pat[i] == '\0')

                                                {

                                                                flag = 1;

                                                                for(k=0; rep[k]!='\0'; k++, j++)

                                                                {

                                                                                ans[j] = rep[k];

                                                                }

                                                                i = 0;

                                                                c = m;

                                                }

                                }

                                else

                                {

                                                ans[j]= str[c];

                                                j++;

                                                c++;

                                                m=c;

                                                i=0;

                                }

                }

                ans[j]='\0';

}

void main()

{

                printf("\nEnter the main string:");

                gets(str);

                printf("\nEnter the pat string:");

                gets(pat);

                printf("\nEnter the replace string:");

                gets(rep);

**stringmatch();**

                if(flag == 1)

                                printf("\nResultant string is %s", ans);

                else

                                printf("\nPattern string is not found");

}

Enter the main string:    **hello aabhii howaab**

Enter the pat string:       **aab**

Enter the replace string:                               **klmno**

Resultant string is: hello **klmno**hii how**klmno**

1. Design, Develop and Implement a Program in C to create a structure to store the name, account number and balance of customers (more than 10) and store their information.

1 - Write a function to print the names of all the customers having balance less than $200.

2 - Write a function to add $100 in the balance of all the customers having more than $1000 in their balance and then print the incremented value of their balance.

#include<stdio.h>

struct bank\_customer{

char name[20];

char accno[50];

float bal;

}s[3];

void printlessthan200(struct bank\_customer s);

void add100(struct bank\_customer s);

void display(struct bank\_customer s);

int main()

{

int i;

for(i=0;i<3;i++)

{

scanf("%s%f%s",s[i].name,&s[i].bal,s[i].accno);

}

/\* for(i=0;i<3;i++)

{

printf("%s%s%f",s[i].name,s[i].accno,s[i].bal);

}\*/

printlessthan200(s);

add100(s);

return 0;

}

void printlessthan200(struct bank\_customer s[])

{

int i;

for(i=0;i<3;i++)

{

if(s[i].bal<200)

{

printf("\nless than 200 %s",s[i].name);

}

}

}

void add100(struct bank\_customer s[])

{

int i;

for(i=0;i<3;i++)

{

if(s[i].bal>1000)

{

s[i].bal=s[i].bal+100;

}

}

display(s);

}

void display(struct bank\_customer s[])

{

int i;

for(i=0;i<3;i++)

{

printf("\n%3s %3s %3f",s[i].name,s[i].accno,s[i].bal);

}

}

**4. Design, Develop and Implement 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 operations**

#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");

            }

}

***Output:***

~~~~~~Menu~~~~~~ :

=>1.Push an Element to Stack and Overflow demo

=>2.Pop an Element from Stack and Underflow demo

=>3.Palindrome demo

=>4.Display

=>5.Exit

Enter your choice: **1**

**Enter an element to be pushed: 11**

~~~~~~Menu~~~~~~ :

=>1.Push an Element to Stack and Overflow demo

=>2.Pop an Element from Stack and Underflow demo

=>3.Palindrome demo

=>4.Display

=>5.Exit

Enter your choice: **1**

**Enter an element to be pushed: 12**

~~~~~~Menu~~~~~~ :

=>1.Push an Element to Stack and Overflow demo

=>2.Pop an Element from Stack and Underflow demo

=>3.Palindrome demo

=>4.Display

=>5.Exit

Enter your choice: **1**

**Enter an element to be pushed: 13**

~~~~~~Menu~~~~~~ :

=>1.Push an Element to Stack and Overflow demo

=>2.Pop an Element from Stack and Underflow demo

=>3.Palindrome demo

=>4.Display

=>5.Exit

Enter your choice: **1**

**Enter an element to be pushed: 14**

~~~~~~Menu~~~~~

=>1.Push an Element to Stack and Overflow demo

=>2.Pop an Element from Stack and Underflow demo

=>3.Palindrome demo

=>4.Display

=>5.Exit

Enter your choice:**1**

**Enter an element to be pushed: 15**

~~~~~~Menu~~~~~

=>1.Push an Element to Stack and Overflow demo

=>2.Pop an Element from Stack and Underflow demo

=>3.Palindrome demo

=>4.Display

=>5.Exit

1. Design, Develop and Implement a Program in C for converting an Infix Expression to Postfix Expression. Program should support for both parenthesized and free parenthesized expressions with the operators: +, -, \*, /, % (Remainder), ^ (Power) and alphanumeric operands.

#include<stdio.h>

int TOP=-1;

char STACK[20];

/\*function to push an item into the stack \*/

void PUSH(char x)

{

TOP++;

STACK[TOP]=x;

}

/\*function to pop an item from the stack \*/

char POP()

{

char temp;

temp=STACK[TOP];

TOP--;

return temp;

}

int PREC(char sym)

{

int p;

switch(sym)

{

case '^': p=3;

break;

case '/': case '\*':

p=2;

break;

case '+': case '-':

p=1;

break;

case '(':case ')':

p=0;

break;

case '#':

p=-1;

break;

}

return(p);

}

int main()

{

char POST[25],INF[25],sym,temp;

inti,pos=0;

printf("Enter a Infix expression \n");

scanf("%s",INF);

PUSH('#');

/\* length of the string POST\*/

intlen=0;

while(INF[len]!='\0')

len++;

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

{

sym=INF[i];

switch(sym)

{

case '(': PUSH(sym);

break;

case ')': temp=POP();

while(temp!='(')

{

POST[pos++]=temp;

temp=POP();

}

break;

case '+':case '-':case '\*':case '/':case '^':

while(PREC(STACK[TOP])>=PREC(sym))

{

temp=POP();

POST[pos++]=temp;

}

PUSH(sym);

break;

default: POST[pos++]=sym;

break;

}

}

while(TOP>0)

{

temp=POP();

POST[pos++]=temp;

}

POST[pos]='\0';

printf("The Postfix Expression : %s",POST);

return 0;

}

**OUTPUT:**

**Enter an INFIX expression**

**(A+B)\*(C-D)**

**AB+CD-\***

6. Design, Develop and Implement a Program in C for the following Stack Applications a. Evaluation of Postfix expression with single digit operands and operators: +, -, \*, /, %, ^.

b. Solving Tower of Hanoi problem with n disks.

#include<stdio.h>

int STACK[20],TOP=-1;

void push(int item)

{

TOP++;

STACK[TOP]=item;

}

int pop()

{

int item;

item=STACK[TOP];

TOP--;

return item;

}

main()

{

char POST[25],sym;

int op1,op2,i;

printf("\n \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_");

printf("\n EVALUATION OF POSTFIX EXPRESSION");

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

printf("\n Enter Postfix expression: ");

scanf("%s",POST);

// printf("%d",'0');

/\* length of the string POST\*/

int len=0;

while(POST[len]!='\0')

len++;

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

{

sym=POST[i];

switch(sym)

{

case '+':

op2=pop();

op1=pop();

push(op1+op2);

break;

case '-':

op2=pop();

op1=pop();

push(op1-op2);

break;

case '\*':

op2=pop();

op1=pop();

push(op1\*op2);

break;

case '/':

op2=pop();

op1=pop();

push(op1/op2);

break;

case '^':

op2=pop();

op1=pop();

push(op1^op2);

break;

default:

push(sym-‘0’);

break;

}

}

printf("\n The value for the given postfix expression %d", pop());

}

**OUTPUT:**

Enter Postfix expression

5    1    2    +    4    \*    +    3    -

14

**PART-B**

7. Design, Develop and Implement 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 to 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

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

#include <stdio.h>

#define SIZE 5

int items[SIZE];

int front = -1, rear =-1;

int isFull()

{

if( (front == rear + 1) || (front == 0 && rear == SIZE-1))

return 1;

return 0;

}

int isEmpty()

{

if(front == -1)

return 1;

return 0;

}

void insert()

{

int element;

if(isFull()) printf("\n Queue is full!! \n");

else

{

printf("enter element");

scanf("%d",&element);

if(front == -1) front = 0;

rear = (rear + 1) % SIZE;

items[rear] = element;

printf("\n Inserted -> %d", element);

}

}

int delete()

{

int element;

if(isEmpty()) {

printf("\n Queue is empty !! \n");

return(-1);

}

else {

element = items[front];

if (front == rear){

front = -1;

rear = -1;

} /\* Q has only one element, so we reset the queue after dequeing it. ? \*/

else {

front = (front + 1) % SIZE;

}

printf("\n Deleted element -> %d \n", element);

return(element);

}

}

void display()

{

int i;

if(isEmpty()) printf(" \n Empty Queue\n");

else

{

printf("\n Front -> %d ",front);

printf("\n Items -> ");

for( i = front; i!=rear; i=(i+1)%SIZE) {

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

}

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

printf("\n Rear -> %d \n",rear);

}

}

int main()

{

int ch;

while(1)

{

printf("enter ch");

scanf("%d",&ch);

switch(ch)

{

case 1:insert();

break;

case 2:delete();

break;

case 3:display();

break;

default:exit(0);

}

}

return 0;}

***Output:***  
~~Main Menu~~  
==> 1. Insertion and Overflow Demo  
==> 2. Deletion and Underflow Demo  
==> 3. Display  
==> 4. Exit  
Enter Your Choice: **1**  
  
**Enter the element to be inserted: A**  
  
~~Main Menu~~  
==> 1. Insertion and Overflow Demo  
==> 2. Deletion and Underflow Demo  
==> 3. Display  
==> 4. Exit  
Enter Your Choice: **1**  
**Enter the element to be inserted: B**  
  
~~Main Menu~~  
==> 1. Insertion and Overflow Demo  
==> 2. Deletion and Underflow Demo  
==> 3. Display  
==> 4. Exit  
Enter Your Choice: **1**  
**Enter the element to be inserted: C**  
  
~~Main Menu~~  
==> 1. Insertion and Overflow Demo  
==> 2. Deletion and Underflow Demo  
==> 3. Display  
==> 4. Exit  
Enter Your Choice: **1**

**Enter the element to be inserted: D**  
**~~Circular Queue Overflow~~**  
  
~~Main Menu~~  
==> 1. Insertion and Overflow Demo  
==> 2. Deletion and Underflow Demo  
==> 3. Display  
==> 4. Exit  
Enter Your Choice: **3**  
  
**Circular Queue contents are:**  
**Front[0]-> A B C <-[2]Rear**  
  
~~Main Menu~~  
==> 1. Insertion and Overflow Demo  
==> 2. Deletion and Underflow Demo  
==> 3. Display  
==> 4. Exit  
Enter Your Choice: **2**  
  
**Deleted element from the queue is: A**  
  
~~Main Menu~~  
==> 1. Insertion and Overflow Demo  
==> 2. Deletion and Underflow Demo  
==> 3. Display  
==> 4. Exit  
Enter Your Choice: **3**

**Circular Queue contents are:**  
**Front[1]-> B C <-[2]Rear**  
  
~~Main Menu~~  
==> 1. Insertion and Overflow Demo  
==> 2. Deletion and Underflow Demo  
==> 3. Display  
==> 4. Exit  
Enter Your Choice: **1**  
  
**Enter the element to be inserted: E**  
  
~~Main Menu~~  
==> 1. Insertion and Overflow Demo  
==> 2. Deletion and Underflow Demo  
==> 3. Display  
==> 4. Exit  
Enter Your Choice: **3**

8. 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

a. Create a SLL of N Students Data by using front insertion.

b. Display the status of SLL and count the number of nodes in it

c. Perform Insertion / Deletion at End of SLL

d. Perform Insertion / Deletion at Front of SLL(Demonstration of stack)

e. Exit

#include<stdio.h>

#include<stdlib.h>

struct node

{

char name[100],usn[100],branch[100];

int sem,phonenum;

struct node \*link;

};

typedef struct node \*NODE;

NODE start = NULL;

int count=0;

void createatend()

{

NODE snode,cur;

snode=(NODE)malloc(sizeof(NODE));

printf("enter name,usn,branch,semester and phone number ");

scanf("%s%s%s%d%d",snode->name,snode->usn,snode->branch,&snode->sem,&snode->phonenum);

snode->link=NULL;

if(snode==NULL)

printf("error");

if(start==NULL)

start=snode;

else

{

cur=start;

while(cur->link!=NULL)

cur=cur->link;

cur->link=snode;

}

count++;

}

void createatfront()

{

NODE snode,cur;

snode=(NODE)malloc(sizeof(NODE));

printf("enter name,usn,branch,semester and phone number ");

scanf("%s%s%s%d%d",snode->name,snode->usn,snode->branch,&snode->sem,&snode->phonenum);

snode->link=NULL;

if(snode==NULL)

printf("error");

if(start==NULL)

start=snode;

else

{

snode->link=start;

start=snode;

}

count++;

}

void deleteatend()

{

NODE cur,prev;

if(start==NULL)

printf("no elments");

else if(start->link==NULL)

{

free(start);

count--;

}

else

{

cur=start;

while(cur->link!=NULL)

{

prev=cur;

cur=cur->link;

}

free(cur);

prev->link=NULL;

count--;

}

}

void deleteatfront()

{

NODE cur;

if(start==NULL)

printf("no elements in list");

else

{

cur=start;

start=start->link;

free(cur);

count--;

}

}

void display()

{

NODE cur;

cur=start;

printf("number of elements in list %d",count);

while(cur!=NULL)

{

printf("%s%s%s%d%d",cur->name,cur->usn,cur->branch,cur->sem,cur->phonenum);

cur=cur->link;

}

}

void stackdemo()

{

int ch;

while(1)

{

printf("enter ch");

scanf("%d",&ch);

switch(ch)

{

case 1:createatfront();

break;

case 2:deleteatfront();

break;

case 3:return;

}}}

main()

{

int ch;

while(1)

{

printf("enter ch");

scanf("%d",&ch);

switch(ch)

{

case 1:createatend();

break;

case 2:createatfront();

break;

case 3:deleteatfront();

break;

case 4:deleteatend();

break;

case 5:display();

break;

case 6:stackdemo();

break;

default: printf("invalid choice");

}

}}

Output:

~~~Menu~~~

Enter your choice for SLL operation

1:Create SLL of Student Nodes

2:DisplayStatus

3:InsertAtEnd

4:DeleteAtEnd

5:Stack Demo using SLL(Insertion and Deletion at Front)

6:Exit

Enter your choice:1

Enter the no of students: 3

Enter the usn,Name,Branch, sem,PhoneNo of the student:

111

aaa

cs

1

111111

Enter the usn,Name,Branch, sem,PhoneNo of the student:

222

bbb

ec

2

222222

Enter the usn,Name,Branch, sem,PhoneNo of the student:

333

ccc

ec

3

333333

~~~Menu~~~

Enter your choice for SLL operation

1:Create SLL of Student Nodes

2:DisplayStatus

3:InsertAtEnd

4:DeleteAtEnd

5:Stack Demo using SLL(Insertion and Deletion at Front)

6:Exit

Enter your choice:2

The contents of SLL:

||1|| USN:333| Name:ccc| Branch:ec| Sem:3| Ph:333333|

||2|| USN:222| Name:bbb| Branch:ec| Sem:2| Ph:222222|

||3|| USN:111| Name:aaa| Branch:cs| Sem:1| Ph:111111|

No of student nodes is 3

~~~Menu~~~

Enter your choice for SLL operation

1:Create SLL of Student Nodes

2:DisplayStatus

3:InsertAtEnd

4:DeleteAtEnd

5:Stack Demo using SLL(Insertion and Deletion at Front)

6:Exit

Enter your choice:3

Enter the usn,Name,Branch, sem,PhoneNo of the student:

444

ddd

ec

4

444444

9. 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

a. Create a DLL of N Employees Data by using end insertion.

b. Display the status of DLL and count the number of nodes in it

c. Perform Insertion and Deletion at End of DLL

d. Perform Insertion and Deletion at Front of DLL

e. Demonstrate how this DLL can be used as Double Ended Queue.

f. Exit

#include<stdio.h>

#include<stdlib.h>

struct node

{

struct node \*prev;

int ssn;

char name[100],dept[100],desg[100];

float phnum,sal;

struct node \*next;

};

typedef struct node \*NODE;

NODE start = NULL;

int count=0;

void insertatlast()

{

NODE snode,temp;

snode=(NODE)malloc(sizeof(NODE));

printf("enter ssn,name,dept,desg,sal,phnum");

scanf("%d%s%s%s%f%f",&snode->ssn,snode->name,snode->dept,snode->desg,&snode->sal,&snode->phnum);

snode->prev=NULL;

snode->next=NULL;

if(snode==NULL)

printf("error");

if(start==NULL)

start=snode;

else

{

temp=start;

while(temp->next!=NULL)

temp=temp->next;

temp->next=snode;

snode->prev=temp;

}

count++;

}

void insertatfront()

{

NODE snode,cur;

snode=(NODE)malloc(sizeof(NODE));

printf("enter ssn,name,dept,desg,sal,phnum");

scanf("%d%s%s%s%f%f",&snode->ssn,snode->name,snode->dept,snode->desg,&snode->sal,&snode->phnum);

snode->next=NULL;

snode->prev=NULL;

if(snode==NULL)

printf("error");

if(start==NULL)

start=snode;

else

{

snode->next=start;

start->prev=snode;

start=snode;

}

count++;

}

void deleteatend()

{

NODE temp,prev;

if(start==NULL)

printf("no elments");

if(start->next==NULL)

{

free(start);

count--;

}

else

{

temp=start;

while(temp->next!=NULL)

{

prev=temp;

temp=temp->next;

}

free(temp);

prev->next=NULL;

count--;

}

}

void deleteatfront()

{

NODE temp;

if(start==NULL)

printf("no elements in list");

else if(start->next==NULL)

{

free(start);

count--;

}

else

{

temp=start;

start=start->next;

start->prev=NULL;

free(temp);

count--;

}

}

void display()

{

NODE temp;

temp=start;

printf("number of elemnts in list %d",count);

while(temp!=NULL)

{

printf("%d%s%s%s%f%f",temp->ssn,temp->name,temp->dept,temp->desg,temp->sal,temp->phnum);

temp=temp->next;

}

}

void dequeuedemo()

{

int ch;

printf(“enter ch”);

scanf(“%d”,&ch);

switch(ch)

{

case 1:insertatlast();

break;

case 2:insertatfront();

break;

case 3:deleteatend();

break;

case 4:deleteatfront();

break;

case 5:return;

}}

main()

{

int ch;

while(1)

{

printf("enter ch");

scanf("%d",&ch);

switch(ch)

{

case 1:insertatlast();

break;

case 2:insertatfront();

break;

case 3:deleteatend();

break;

case 4:deleteatfront();

break;

case 5:display();

break;

case 6:dequeuedemo();

break;

default: printf("invalid choice");

}

}

}

Output:

~~~Menu~~~

1:Create DLL of Employee Nodes

2:DisplayStatus

3:InsertAtEnd

4:DeleteAtEnd

5:InsertAtFront

6:DeleteAtFront

7:Double Ended Queue Demo using DLL

8:Exit

Please enter your choice: 1

Enter the no of Employees: 2

Enter the ssn,Name,Department,Designation,Salary,PhoneNo of the employee:

111

aaa

dept1

des1

1000

11111

Enter the ssn,Name,Department,Designation,Salary,PhoneNo of the employee:

222

bbb

dept2

des2

2000

22222

10. Using circular representation for a polynomial, design, develop, and execute a program in C to accept two polynomials, add them, and then print the resulting polynomial.

#include<stdio.h>

#include<stdlib.h>

struct node

{

int coef;

int pow;

struct node\*link;

};

struct node\*h1,\*h2,\*h3;

void create(struct node\*h)

{

int i=1,c,p,X;

struct node\*temp,\*cur;

while(i)

{

printf("enter the coef n pow\n");

scanf("%d%d",&c,&p);

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

temp->coef=c;

temp->pow=p;

cur=h->link;

while(cur->link!=h)

{

cur=cur->link;

}

cur->link=temp;

temp->link=h;

printf("Enter 0 to Quit and 1 to Continue\n");

scanf("%d",&i);

}

}

void display(struct node\*h)

{

struct node\*cur;

cur=h->link;

while(cur->link!=h)

{

printf("%dx^%d+",cur->coef,cur->pow);

cur=cur->link;

}

printf("%dx^%d",cur->coef,cur->pow);

}

void final(int c,int p)

{

struct node\*cur,\*temp;

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

temp->pow=p;

temp->coef=c;

cur=h3->link;

while(cur->link!=h3)

{

cur=cur->link;

}

cur->link=temp;

temp->link=h3;

}

void add(struct node\*h1,struct node\*h2)

{

struct node\*a,\*b;

a=h1->link;

b=h2->link;

while(a!=h1 && b!=h2)

{

if((a->pow) > (b->pow))

{

final(a->coef,a->pow);

a=a->link;

}

else if ((a->pow) > (b->pow))

{

final(b->coef,b->pow);

b=b->link;

}

else

{

final(a->coef+b->c

oef,a->pow);

a=a->link;

b=b->link;

}

}

while(a!=h1)

{

final(a->coef,a->pow);

a=a->link;

}

while(b!=h2)

{

final(b->coef,b->pow);

b=b->link;

}

}

main()

{

h1=(struct node\*)malloc(sizeof (struct node));

h2=(struct node\*)malloc(sizeof (struct node));

h3=(struct node\*)malloc(sizeof (struct node));

h1->link=h1;

h2->link=h2;

h3->link=h3;

printf("enter first poly\n");

create(h1);

printf("enter second poly");

create(h2);

printf("first poly\n");

display(h1);

printf("second poly\n");

display(h2);

printf("third poly\n");

add(h1,h2);

display(h3);

}

11. 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>

struct BST

{

int data;

struct BST \*lchild;

struct BST \*rchild;

};

typedef struct BST \* NODE;

NODE create()

{

NODE temp;

temp = (NODE) malloc(sizeof(struct BST));

printf("\nEnter The value: ");

scanf("%d", &temp->data);

temp->lchild = NULL;

temp->rchild = NULL;

return temp;

}

void insert(NODE root, NODE newnode);

void inorder(NODE root);

void preorder(NODE root);

void postorder(NODE root);

void search(NODE root);

void insert(NODE root, NODE newnode)

{

/\*Note: if newnode->data == root->data it will be skipped. No duplicate nodes are allowed \*/

if (newnode->data < root->data)

{

if (root->lchild == NULL)

root->lchild = newnode;

else

Insert(root->lchild, newnode);

}

if (newnode->data > root->data)

{

if (root->rchild == NULL)

root->rchild = newnode;

else

insert(root->rchild, newnode);

}

}

void search(NODE root)

{

int key;

NODE cur;

if(root == NULL)

{

printf("\nBST is empty.");

return;

}

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

scanf("%d", &key);

cur = root;

while (cur != NULL)

{

if (cur->data == key)

{

printf("\nKey element is present in BST");

return;

}

if (key < cur->data)

cur = cur->lchild;

else

cur = cur->rchild;

}

printf("\nKey element is not found in the BST");

}

void inorder(NODE root)

{

if(root != NULL)

{

inorder(root->lchild);

printf("%d ", root->data);

inorder(root->rchild);

}

}

void preorder(NODE root)

{

if (root != NULL)

{

printf("%d ", root->data);

preorder(root->lchild);

preorder(root->rchild);

}

}

void postorder(NODE root)

{

if (root != NULL)

{

postorder(root->lchild);

postorder(root->rchild);

printf("%d ", root->data);

}

}

void main()

{

int ch, key, val, i, n;

NODE root = NULL, newnode;

while(1)

{

printf("\n~~~~BST MENU~~~~");

printf("\n1.Create a BST");

printf("\n2.Search");

printf("\n3.BST Traversals: ");

printf("\n4.Exit");

printf("\nEnter your choice: ");

scanf("%d", &ch);

switch(ch)

{

case 1: printf("\nEnter the number of elements: ");

scanf("%d", &n);

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

{

newnode = create();

if (root == NULL)

root = newnode;

else

insert(root, newnode);

}

break;

case 2: if (root == NULL)

printf("\nTree Is Not Created");

else

{

printf("\nThe Preorder display : ");

preorder(root);

printf("\nThe Inorder display : ");

inorder(root);

printf("\nThe Postorder display : ");

postorder(root);

}

break;

case 3: search(root);

break;

case 4: exit(0);

}

}

}

Output:

~~~~BST MENU~~~~

1.Create a BST

2.Search

3.BST Traversals:

4.Exit

Enter your choice: 1

Enter the number of elements: 12

Enter The value: 6

Enter The value: 9

Enter The value: 5

Enter The value: 2

Enter The value: 8

Enter The value: 15

Enter The value: 24

Enter The value: 14

Enter The value: 7

Enter The value: 8

Enter The value: 5

Enter The value: 2

~~~~BST MENU~~~~

1.Create a BST

2.Search

3.BST Traversals:

4.Exit

Enter your choice: 3

The Preorder display: 6 5 2 9 8 7 15 14 24

The Inorder display: 2 5 6 7 8 9 14 15 24

The Postorder display: 2 5 7 8 14 24 15 9 6

~~~~BST MENU~~~~

1.Create a BST

2.Search

3.BST Traversals:

4.Exit

Enter your choice: 2

Enter Element to be searched: 66

Key element is not found in the BST

~~~~BST MENU~~~~

1.Create a BST

2.Search

3.BST Traversals:

4.Exit

Enter your choice: 2

Enter Element to be searched: 14

Key element is present in BST

~~~~BST MENU~~~~

1.Create a BST

2.Search

3.BST Traversals:

4.Exit

12. Construct a dictionary of key-value pairs using Tree and search for a value matching a key.

#include <stdio.h>

#include <stdlib.h>

#define MAX 50

struct Node

{

char data;

unsigned isEndOfString: 1;

struct Node \*left, \*eq, \*right;

};

// A utility function to create a new ternary search tree node

struct Node\* newNode(char data)

{

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

temp->data = data;

temp->isEndOfString = 0;

temp->left = temp->eq = temp->right = NULL;

return temp;

}

// Function to insert a new word in a Ternary Search Tree

void insert(struct Node\*\* root, char \*word)

{

// Base Case: Tree is empty

if (!(\*root))

\*root = newNode(\*word);

// If current character of word is smaller than root's character,

// then insert this word in left subtree of root

if ((\*word) < (\*root)->data)

insert(&( (\*root)->left ), word);

// If current character of word is greate than root's character,

// then insert this word in right subtree of root

else if ((\*word) > (\*root)->data)

insert(&( (\*root)->right ), word);

// If current character of word is same as root's character,

else

{

if (\*(word+1))

insert(&( (\*root)->eq ), word+1);

// the last character of the word

else

(\*root)->isEndOfString = 1;

}

}

// A recursive function to traverse Ternary Search Tree

void traverseTSTUtil(struct Node\* root, char\* buffer, int depth)

{

if (root)

{

// First traverse the left subtree

traverseTSTUtil(root->left, buffer, depth);

// Store the character of this node

buffer[depth] = root->data;

if (root->isEndOfString)

{

buffer[depth+1] = '\0';

printf( "%s\n", buffer);

}

// Traverse the subtree using equal pointer (middle subtree)

traverseTSTUtil(root->eq, buffer, depth + 1);

// Finally Traverse the right subtree

traverseTSTUtil(root->right, buffer, depth);

}

}

// The main function to traverse a Ternary Search Tree.

// It mainly uses traverseTSTUtil()

void traverseTST(struct Node\* root)

{

char buffer[MAX];

traverseTSTUtil(root, buffer, 0);

}

// Function to search a given word in TST

int searchTST(struct Node \*root, char \*word)

{

if (!root)

return 0;

if (\*word < (root)->data)

return searchTST(root->left, word);

else if (\*word > (root)->data)

return searchTST(root->right, word);

else

{

if (\*(word+1) == '\0')

return root->isEndOfString;

return searchTST(root->eq, word+1);

}

}

// Driver program to test above functions

int main()

{

struct Node \*root = NULL;

insert(&root, "cat");

insert(&root, "cats");

insert(&root, "up");

insert(&root, "bug");

printf("Following is traversal of ternary search tree\n");

traverseTST(root);

printf("\nFollowing are search results for cats, bu and cat respectively\n");

searchTST(root, "cats")? printf("Found\n"): printf("Not Found\n");

searchTST(root, "bu")? printf("Found\n"): printf("Not Found\n");

searchTST(root, "cat")? printf("Found\n"): printf("Not Found\n");

return 0;

}

OUTPUT:

Following is traversal of ternary search tree

bug

cat

cats

up

Following are search results for cats, bu and cat respectively

Found

Not Found

Found