1) Array Insertion, Deletion, Display by Dynamic Memory Allocation

#include<stdio.h>

#include<stdlib.h>

int insertion(int \*a, int \*n)

{

int i,ele,pos;

printf("enter the position where you want to insert an element: ");

scanf("%d",&pos);

printf("enter element to insert: ");

scanf("%d",&ele);

if(pos>\*n) {

printf("invalid input");

}

else {

\*n=\*n+1;

a = (int\*)realloc(a,(\*n)\*sizeof(int));

for(i=\*n-1;i>=pos;i--) {

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

}

a[pos]=ele;

}

}

int display(int \*a, int \*n)

{

int i;

for(i=0;i<\*n;i++) {

printf("%d\t",\*(a+i));

}

}

int deletion(int \*a, int \*n)

{

int pos,i;

printf("enter the position where you want to delete an element: ");

scanf("%d",&pos);

if(pos>=\*n) {

printf("invalid input");

}

else {

\*n=\*n-1;

for(i=pos;i<=\*n;i++) {

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

}

a = (int\*)realloc(a,(\*n)\*sizeof(int));

}

}

int main()

{

int n,i,ch,c=0;

printf("enter the size of the array: ");

scanf("%d",&n);

int\* a=(int\*)malloc(n\*sizeof(int));

printf("enter the elements of the array: ");

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

scanf("%d",(a+i));

}

while(1) {

printf("\nPress 1 to insert");

printf("\nPress 2 to delete");

printf("\nPress 3 to display");

printf("\nPress 4 to exit");

printf("\nenter your choice: ");

scanf("%d",&ch);

switch(ch)

{

case 1: insertion(a,&n);

break;

case 2: deletion(a,&n);

break;

case 3: display(a,&n);

break;

case 4: exit(0);

default: printf("invalid choice");

}

}

}

Output

enter the size of the array: 4

enter the elements of the array: 1

2

3

4

Press 1 to insert

Press 2 to delete

Press 3 to display

Press 4 to exit

enter your choice: 1

enter the position where you want to insert an element: 5

enter element to insert: 7

invalid input

Press 1 to insert

Press 2 to delete

Press 3 to display

Press 4 to exit

enter your choice: 1

enter the position where you want to insert an element: 3

enter element to insert: 2

Press 1 to insert

Press 2 to delete

Press 3 to display

Press 4 to exit

enter your choice: 3

1 2 3 2 4

Press 1 to insert

Press 2 to delete

Press 3 to display

Press 4 to exit

enter your choice: 2

enter the position where you want to delete an element: 2

Press 1 to insert

Press 2 to delete

Press 3 to display

Press 4 to exit

enter your choice: 3

1 2 2 4

Press 1 to insert

Press 2 to delete

Press 3 to display

Press 4 to exit

enter your choice: 4

Process returned 0 (0x0) execution time : 365.614 s

P

4

Press 1 to insert

Press 2 to delete

Press 3 to display

Press 4 to exit

enter your choice: 1

enter the position where you want to insert an element: 5

enter element to insert: 7

invalid input

Press 1 to insert

Press 2 to delete

Press 3 to display

Press 4 to exit

enter your choice: 1

enter the position where you want to insert an element: 3

enter element to insert: 2

Press 1 to insert

Press 2 to delete

Press 3 to display

Press 4 to exit

enter your choice: 3

1 2 3 2 4

Press 1 to insert

Press 2 to delete

Press 3 to display

Press 4 to exit

enter your choice: 2

enter the position where you want to delete an element: 2

Press 1 to insert

Press 2 to delete

Press 3 to display

Press 4 to exit

enter your choice: 3

1 2 2 4

Press 1 to insert

Press 2 to delete

Press 3 to display

Press 4 to exit

enter your choice: 4

Process returned 0 (0x0) execution time : 365.614 s

P

2) Insert a node at the first and last position and Delete a node at the first and last position of a Singly Linked List

#include<stdio.h>

#include<stdlib.h>

typedef struct node

{

int val;

struct node \*next;

}node;

node\* create()

{

int item;

node \*ptr;

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

printf("enter data: ");

scanf("%d",&item);

ptr->val = item;

ptr->next = NULL;

return ptr;

}

void insert\_first(node \*\*start)

{

node \*temp;

temp = create();

if((\*start) == NULL) (\*start) = temp;

else {

temp->next = (\*start);

(\*start) = temp;

}

}

void insert\_last(node \*\*start)

{

node \*ptr,\*temp;

temp = create();

if((\*start) == NULL) (\*start) = ptr;

else{

ptr = (\*start);

while(ptr->next!=NULL) {

ptr = ptr->next;

}

ptr->next = temp;

}

}

void display(node \*\*start)

{

node \*ptr,\*temp;

ptr = (\*start);

if((\*start) == NULL) printf("no node present in the Linked List");

else{

printf("\n");

while(ptr!=NULL) {

printf("%d ",ptr->val);

ptr = ptr->next;

}

printf("\n");

}

}

void delete\_first(node \*\*start)

{

node \*ptr,\*temp;

if((\*start) == NULL) printf("deletion is not possible");

else{

temp = (\*start);

(\*start) = (\*start)->next;

free(temp);

}

}

void delete\_last(node \*\*start)

{

node \*ptr,\*preptr=NULL;

if((\*start) == NULL) printf("deletion is no t possible");

else{

ptr = (\*start);

preptr = (\*start);

while(ptr->next!=NULL)

{

preptr = ptr;

ptr = ptr->next;

}

preptr->next = NULL;

free(ptr);

return;

}

}

int main()

{

node \*start = NULL;

int ch;

while(1) {

printf("\npress 1 to insert at first");

printf("\npress 2 to insert at last");

printf("\npress 3 to display");

printf("\npress 4 to delete first node");

printf("\npress 5 to delete last node");

printf("\npress 6 to exit");

printf("\nenter your choice:");

scanf("%d",&ch);

switch(ch) {

case 1:insert\_first(&start);

break;

case 2:insert\_last(&start);

break;

case 3:display(&start);

break;

case 4:delete\_first(&start);

break;

case 5:delete\_last(&start);

break;

case 6:exit(0);

default:printf("invalid choice");

}

}

}

Output

press 1 to insert at first

press 2 to insert at last

press 3 to display

press 4 to delete first node

press 5 to delete last node

press 6 to exit

enter your choice:1

enter data: 2

press 1 to insert at first

press 2 to insert at last

press 3 to display

press 4 to delete first node

press 5 to delete last node

press 6 to exit

enter your choice:1

enter data: 5

press 1 to insert at first

press 2 to insert at last

press 3 to display

press 4 to delete first node

press 5 to delete last node

press 6 to exit

enter your choice:2

enter data: 6

press 1 to insert at first

press 2 to insert at last

press 3 to display

press 4 to delete first node

press 5 to delete last node

press 6 to exit

enter your choice:2

enter data: 8

press 1 to insert at first

press 2 to insert at last

press 3 to display

press 4 to delete first node

press 5 to delete last node

press 6 to exit

enter your choice:3

5 2 6 8

press 3 to display

press 4 to delete first node

press 5 to delete last node

press 6 to exit

enter your choice:3

2 6

press 1 to insert at first

press 2 to insert at last

press 3 to display

press 4 to delete first node

press 5 to delete last node

press 6 to exit

enter your choice:5

enter your choice:3

no node present in the Linked List

press 1 to insert at first

press 2 to insert at last

press 3 to display

press 4 to delete first node

press 5 to delete last node

press 6 to exit

enter your choice:6

Process returned 0 (0x0) execution time : 12.612 s

Press any key to continue.

3) Insert a node at the first and last position and Delete a node at the first and last position of a Circular Linked List

#include<stdio.h>

#include<stdlib.h>

typedef struct node

{

int val;

struct node \*next;

}node;

node\* create()

{

int item;

node \*ptr;

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

printf("enter data: ");

scanf("%d",&item);

ptr->val = item;

ptr->next = NULL;

return ptr;

}

void insert\_first(node \*\*start)

{

node \*ptr,\*temp;

temp = create();

if((\*start) == NULL) {

(\*start) = temp;

temp->next = (\*start);

}

else {

ptr = (\*start);

while(ptr->next!=(\*start)) {

ptr = ptr->next;

}

ptr->next = temp;

temp->next = (\*start);

(\*start) = temp;

}

}

void insert\_last(node \*\*start)

{

node \*ptr,\*temp;

temp = create();

if((\*start) == NULL) {

(\*start) = temp;

temp->next = (\*start);

}

else{

ptr = (\*start);

while(ptr->next!=(\*start)) {

ptr = ptr->next;

}

ptr->next = temp;

temp->next = (\*start);

}

}

void display(node \*\*start)

{

node \*ptr;

ptr = (\*start);

if(\*start == NULL) printf("no node present in the CLL");

else{

printf("\n");

while(ptr->next!=(\*start)) {

printf("%d ",ptr->val);

ptr = ptr->next;

}

printf("%d ",ptr->val);

printf("\n");

}

}

void delete\_first(node \*\*start)

{

node \*ptr,\*temp;

if((\*start) == NULL) printf("CLL empty");

else if((\*start)->next == (\*start)) {

ptr = (\*start);

(\*start) = NULL;

free(ptr);

}

else {

ptr = (\*start);

while(ptr->next!=(\*start)) {

ptr = ptr->next;

}

temp = (\*start);

(\*start) = (\*start)->next;

ptr->next = (\*start);

free(temp);

}

}

void delete\_last(node \*\*start)

{

node \*ptr,\*temp;

if((\*start) == NULL) printf("CLL empty");

else if((\*start)->next == (\*start)) {

ptr = (\*start);

(\*start) = NULL;

free(ptr);

}

else {

ptr = (\*start);

temp = (\*start);

while(ptr->next!=(\*start)) {

temp = ptr;

ptr = ptr->next;

}

temp->next = (\*start);

free(ptr);

}

}

int main()

{

node \*start = NULL;

int ch;

while(1) {

printf("\npress 1 to insert at first");

printf("\npress 2 to insert at last");

printf("\npress 3 to display");

printf("\npress 4 to delete first node");

printf("\npress 5 to delete last node");

printf("\npress 6 to exit");

printf("\nenter your choice:");

scanf("%d",&ch);

switch(ch) {

case 1:insert\_first(&start);

break;

case 2:insert\_last(&start);

break;

case 3:display(&start);

break;

case 4:delete\_first(&start);

break;

case 5:delete\_last(&start);

break;

case 6:exit(0);

default:printf("invalid choice");

}

}

}

Output

press 1 to insert at first

press 2 to insert at last

press 3 to display

press 4 to delete first node

press 5 to delete last node

press 6 to exit

enter your choice:1

enter data: 2

press 1 to insert at first

press 2 to insert at last

press 3 to display

press 4 to delete first node

press 5 to delete last node

press 6 to exit

enter your choice:1

enter data: 4

press 1 to insert at first

press 2 to insert at last

press 3 to display

press 4 to delete first node

press 5 to delete last node

press 6 to exit

enter your choice:2

enter data: 5

press 1 to insert at first

press 2 to insert at last

press 3 to display

press 4 to delete first node

press 5 to delete last node

press 6 to exit

enter your choice:2

enter data: 8

press 1 to insert at first

press 2 to insert at last

press 3 to display

press 4 to delete first node

press 5 to delete last node

press 6 to exit

enter your choice:3

4 2 5 8

press 1 to insert at first

press 2 to insert at last

press 3 to display

press 4 to delete first node

press 5 to delete last node

press 6 to exit

enter your choice:4

press 1 to insert at first

press 2 to insert at last

press 3 to display

press 4 to delete first node

press 5 to delete last node

press 6 to exit

enter your choice:3

2 5 8

press 1 to insert at first

press 2 to insert at last

press 3 to display

press 4 to delete first node

press 5 to delete last node

press 6 to exit

enter your choice:5

2 5

4) Insert a node at the first and last position and Delete a node at the first and last position of a Doubly Linked List

#include<stdio.h>

#include<stdlib.h>

typedef struct node

{

int val;

struct node \*prev,\*next;

}node;

node \*create()

{

int item;

node \*ptr;

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

printf("enter element: ");

scanf("%d",&item);

ptr->val = item;

ptr->prev = NULL;

ptr->next = NULL;

return ptr;

}

void insert\_first(node \*\*start)

{

node \*ptr, \*temp;

temp = create();

if((\*start) == NULL) (\*start) = temp;

else{

temp->next = (\*start);

(\*start)->prev = temp;

(\*start) = temp;

}

}

void insert\_last(node \*\*start)

{

node \*ptr,\*temp;

temp = create();

if((\*start) == NULL) (\*start) = temp;

else if((\*start)->next == NULL) {

(\*start)->next = temp;

temp->prev = (\*start);

}

else {

ptr = (\*start);

while(ptr->next!=NULL) {

ptr = ptr->next;

}

ptr->next = temp;

temp->prev = ptr;

temp->next = NULL;

}

}

void delete\_first(node \*\*start)

{

node \*ptr,\*temp;

if((\*start) == NULL) printf("\nDLL is empty");

else if((\*start)->next == NULL) {

temp = (\*start);

(\*start) = NULL;

free(temp);

}

else {

temp = (\*start);

(\*start) = (\*start)->next;

(\*start)->prev = NULL;

free(temp);

}

}

void delete\_last(node \*\*start)

{

node \*ptr, \*temp;

if((\*start) == NULL) printf("\nDLL is empty");

else if((\*start)->next == NULL) {

temp = (\*start);

(\*start) = NULL;

free(temp);

}

else {

ptr = (\*start);

while(ptr->next!=NULL) {

ptr = ptr->next;

}

temp = ptr;

ptr->prev->next = NULL;

free(temp);

}

}

void display(node \*\*start)

{

node \*ptr;

ptr = (\*start);

if((\*start) == NULL) printf("DLL empty");

else {

printf("\n");

while(ptr!=NULL) {

printf("%d\t",ptr->val);

ptr = ptr->next;

}

printf("\n");

}

}

int main()

{

node \*start = NULL;

int ch;

while(1) {

printf("\npress 1 to insert at first");

printf("\npress 2 to insert at last");

printf("\npress 3 to display");

printf("\npress 4 to delete first element");

printf("\npress 5 to delete last element");

printf("\npress 6 to exit");

printf("\nenter your choice:");

scanf("%d",&ch);

switch(ch) {

case 1:insert\_first(&start);

break;

case 2:insert\_last(&start);

break;

case 3:display(&start);

break;

case 4:delete\_first(&start);

break;

case 5:delete\_last(&start);

break;

case 6:exit(0);

default:printf("invalid choice");

}

}

}

Output

press 1 to insert at first

press 2 to insert at last

press 3 to display

press 4 to delete first element

press 5 to delete last element

press 6 to search and insert element after a specific value

press 7 to exit

enter your choice:1

Enter element: 2

press 1 to insert at first

press 2 to insert at last

press 3 to display

press 4 to delete first element

press 5 to delete last element

press 6 to search and insert element after a specific value

press 7 to exit

enter your choice:1

Enter element: 5

press 1 to insert at first

press 2 to insert at last

press 3 to display

press 4 to delete first element

press 5 to delete last element

press 6 to search and insert element after a specific value

press 7 to exit

enter your choice:2

Enter element: 4

press 1 to insert at first

press 2 to insert at last

press 3 to display

press 4 to delete first element

press 5 to delete last element

press 6 to search and insert element after a specific value

press 7 to exit

enter your choice:2

Enter element: 8

press 1 to insert at first

press 2 to insert at last

press 3 to display

press 4 to delete first element

press 5 to delete last element

press 6 to search and insert element after a specific value

press 7 to exit

enter your choice:3

5 2 4 8

press 1 to insert at first

press 2 to insert at last

press 3 to display

press 4 to delete first element

press 5 to delete last element

press 6 to search and insert element after a specific value

press 7 to exit

enter your choice:4

press 1 to insert at first

press 2 to insert at last

press 3 to display

press 4 to delete first element

press 5 to delete last element

press 6 to search and insert element after a specific value

press 7 to exit

enter your choice:3

2 4 8

press 1 to insert at first

press 2 to insert at last

press 3 to display

press 4 to delete first element

press 5 to delete last element

press 6 to search and insert element after a specific value

press 7 to exit

enter your choice:5

press 1 to insert at first

press 2 to insert at last

press 3 to display

press 4 to delete first element

press 5 to delete last element

press 6 to search and insert element after a specific value

press 7 to exit

enter your choice:3

2 4

press 1 to insert at first

press 2 to insert at last

press 3 to display

press 4 to delete first element

press 5 to delete last element

press 6 to search and insert element after a specific value

press 7 to exit

enter your choice:4

press 1 to insert at first

press 2 to insert at last

press 3 to display

press 4 to delete first element

press 5 to delete last element

press 6 to search and insert element after a specific value

press 7 to exit

enter your choice:3

DLL empty

press 1 to insert at first

press 2 to insert at last

press 3 to display

press 4 to delete first element

press 5 to delete last element

press 6 to search and insert element after a specific value

press 7 to exit

enter your choice:7

5) Implement Stack using Array

#include<stdio.h>

#include<stdlib.h>

void push(int \*arr, int n, int \*top)

{

int val;

if((\*top) == n-1) printf("\nOVERFLOW!\n");

else {

(\*top)++;

printf("Enter data to PUSH: \n");

scanf("%d",&val);

arr[(\*top)] = val;

}

}

void pop(int \*arr, int n, int \*top)

{

if((\*top) == -1) printf("\nUNDERFLOW!\n");

else {

(\*top)--;

}

}

void display(int \*arr, int \*top)

{

int i;

if((\*top) == -1) printf("\nUNDERFLOW!\n");

else {

for(i=(\*top);i>=0;i--) {

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

}

}

printf("\n");

}

int main()

{

int \*arr,n,top,ch;

printf("Enter size of Stack: ");

scanf("%d",&n);

top = -1;

arr = (int\*)malloc(sizeof(int)\*n);

while(1) {

printf("1->PUSH\n");

printf("2->POP\n");

printf("3->DISPLAY\n");

printf("4->EXIT\n");

printf("Enter your choice: ");

scanf("%d",&ch);

switch(ch) {

case 1: push(arr,n,&top);

break;

case 2: pop(arr,n,&top);

break;

case 3: display(arr,&top);

break;

case 4: exit(0);

break;

default: printf("INVALID CHOICE!");

}

}

}

Output

Enter size of Stack: 5

1->PUSH

2->POP

3->DISPLAY

4->EXIT

Enter your choice: 1

Enter data to PUSH:

2

1->PUSH

2->POP

3->DISPLAY

4->EXIT

Enter your choice: 1

Enter data to PUSH:

5

1->PUSH

2->POP

3->DISPLAY

4->EXIT

Enter your choice: 1

Enter data to PUSH:

8

1->PUSH

2->POP

3->DISPLAY

4->EXIT

Enter your choice: 3

Enter your choice: 1

Enter data to PUSH:

8

1->PUSH

2->POP

3->DISPLAY

4->EXIT

Enter your choice: 3

8

5

2

1->PUSH

2->POP

3->DISPLAY

4->EXIT

Enter your choice: 1

Enter data to PUSH:

4

1->PUSH

2->POP

3->DISPLAY

4->EXIT

Enter your choice: 3

4

8

5

2

1->PUSH

2->POP

3->DISPLAY

4->EXIT

Enter your choice: 2

1->PUSH

2->POP

3->DISPLAY

4->EXIT

Enter your choice: 3

8

5

2

1->PUSH

2->POP

3->DISPLAY

4->EXIT

Enter your choice: 2

1->PUSH

2->POP

3->DISPLAY

4->EXIT

Enter your choice: 3

5

2

1->PUSH

2->POP

3->DISPLAY

4->EXIT

Enter your choice: 2

1->PUSH

2->POP

3->DISPLAY

4->EXIT

Enter your choice: 3

2

1->PUSH

2->POP

3->DISPLAY

4->EXIT

Enter your choice: 2

1->PUSH

2->POP

3->DISPLAY

4->EXIT

Enter your choice: 3

UNDERFLOW!

1->PUSH

2->POP

3->DISPLAY

4->EXIT

Enter your choice: 4

6) Implement Stack using Linked List

#include<stdio.h>

#include<stdlib.h>

typedef struct node

{

int val;

struct node \*prev;

struct node \*next;

}node;

node \*create()

{

node \*temp;

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

printf("Enter element for NEW node: ");

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

temp->prev = NULL;

temp->next = NULL;

return temp;

}

void push(node \*\*start,node \*\*top,node \*\*end)

{

node \*ptr, \*temp;

temp = create();

if((\*start) == NULL)

{

(\*start) = temp;

(\*top) = temp;

(\*end) = temp;

}

else {

temp->next = (\*start);

(\*start)->prev = temp;

(\*start) = temp;

(\*top) = temp;

}

}

void pop(node \*\*start,node \*\*top, node \*\*end)

{

node \*ptr,\*temp;

if((\*start) == NULL) printf("\nSTACK EMPTY!\n");

else if((\*start)->next == NULL) {

temp = (\*start);

(\*start) = NULL;

(\*top) == NULL;

(\*end) == NULL;

free(temp);

}

else {

temp = (\*start);

(\*start) = (\*start)->next;

(\*top) = (\*start);

free(temp);

}

}

void display(node \*\*start,node \*\*top,node \*\*end)

{

node \*ptr;

if((\*start) == NULL) printf("\nSTACK EMPTY!\n");

else {

ptr = (\*start);

while(ptr!=(\*end)) {

printf("%d\n",ptr->val);

ptr = ptr->next;

}

}

}

int main()

{

node \*start = NULL;

node \*top = start;

node \*end;

int ch;

while(1) {

printf("1->PUSH\n");

printf("2->POP\n");

printf("3->DISPLAY\n");

printf("4->EXIT\n");

printf("Enter your choice: ");

scanf("%d",&ch);

switch(ch) {

case 1: push(&start,&top,&end);

break;

case 2: pop(&start,&top,&end);

break;

case 3: display(&start,&top,&end);

break;

case 4: exit(0);

break;

default: printf("\nINVALID CHOICE!\n");

}

}

}

Output

1->PUSH

2->POP

3->DISPLAY

4->EXIT

Enter your choice: 1

Enter element for NEW node: 2

1->PUSH

2->POP

3->DISPLAY

4->EXIT

Enter your choice: 1

Enter element for NEW node: 4

1->PUSH

2->POP

3->DISPLAY

4->EXIT

Enter your choice: 1

Enter element for NEW node: 8

1->PUSH

2->POP

3->DISPLAY

4->EXIT

Enter your choice: 3

8

4

2

1->PUSH

2->POP

3->DISPLAY

4->EXIT

Enter your choice: 2

1->PUSH

2->POP

3->DISPLAY

4->EXIT

Enter your choice: 3

4

2

1->PUSH

2->POP

3->DISPLAY

4->EXIT

Enter your choice: 3

2

1->PUSH

2->POP

3->DISPLAY

4->EXIT

4->EXIT

Enter your choice: 3

STACK EMPTY!

1->PUSH

2->POP

3->DISPLAY

4->EXIT

Enter your choice: 4

7) Postfix evaluation

#include<stdio.h>

#include<stdlib.h>

#include<string.h>

#include<ctype.h>

void push(char x,char \*arr, int \*top)

{

arr[++(\*top)] = x;

}

char pop(char \*arr, int \*top)

{

if((\*top) == -1) return -1;

else return arr[(\*top)--];

}

int main()

{

char \*arr;

char exp[20];

printf("ENTER EXPRESSION: ");

scanf("%s",exp);

char \*e,x;

e = exp;

int top = -1;

int a,b,c,d,z=0,y=0,m;

arr = (char\*)malloc(strlen(exp)\*sizeof(char));

if(\*e != '\0') {

if(\*e == '+' || \*e == '-' || \*e == '\*' || \*e == '/' || \*e == '^' || \*e == '$') {

printf("NOT A POSTFIX!");

exit(0);

}

}

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

if(isdigit(\*e)) {

c = (\*e) - 48;

push(c,arr,&top);

z++;

}

else {

y++;

a = pop(arr,&top);

b = pop(arr,&top);

switch(\*e) {

case '+': d = a + b;

break;

case '-': d = a - b;

break;

case '\*': d = a \* b;

break;

case '/': d = (float)a / b;

break;

case '^':case '$':

m = 1;

while(b!=0) {

m = m \* a;

b--;

}

d = m;

break;

default: printf("INVALID OPERATOR FOUND!");

}

push(d,arr,&top);

}

e++;

}

if(z == y+1) {

printf("RESULT = %d",pop(arr,&top));

}

else printf("NOT A POSTFIX!");

}

Output

ENTER EXPRESSION: 1536\*+-

RESULT = 22

8) Tower of Hanoi

#include<stdio.h>

void Tower\_of\_Hanoi(int n, char BEG, char END, char AUX)

{

if(n == 1)

{

printf("\n DISC 1 FROM %c TO %c",BEG,END);

return;

}

else {

Tower\_of\_Hanoi(n-1,BEG,AUX,END);

printf("\n DISC %d FROM %c TO %c",n,BEG,END);

Tower\_of\_Hanoi(n-1,AUX,END,BEG);

}

}

int main()

{

int n;

printf("ENTER NUMBER OF DISCS: ");

scanf("%d",&n);

Tower\_of\_Hanoi(n,'A','C','B');

}

Output

ENTER NUMBER OF DISCS: 4

DISC 1 FROM A TO B

DISC 2 FROM A TO C

DISC 1 FROM B TO C

DISC 3 FROM A TO B

DISC 1 FROM C TO A

DISC 2 FROM C TO B

DISC 1 FROM A TO B

DISC 4 FROM A TO C

DISC 1 FROM B TO C

DISC 2 FROM B TO A

DISC 1 FROM C TO A

DISC 3 FROM B TO C

DISC 1 FROM A TO B

DISC 2 FROM A TO C

DISC 1 FROM B TO C

9) Infix to Postfix

#include<stdio.h>

#include<stdlib.h>

#include<string.h>

#include<ctype.h>

void push(char x, char \*arr, int \*top)

{

arr[++(\*top)] = x;

}

char pop(char \*arr, int \*top)

{

if((\*top) == -1) return -1;

else return arr[(\*top)--];

}

int precedence(char x)

{

if(x == '(') return 0;

if(x == '+' || x == '-') return 1;

if(x == '\*' || x == '/') return 2;

if(x == '^' || x == '$') return 3;

return 0;

}

int main()

{

char \*arr;

char exp[] = "A+(B/C-(D\*E^F)+G)\*H";

char \*e,x;

e = exp;

int top = -1;

arr = (char\*)malloc(strlen(exp)\*sizeof(char));

if((\*e !='\0') && (\*e == '+' || \*e == '-' || \*e == '\*' || \*e == '/' || \*e == '^' || \*e == '$')) {

printf("INVALID EXP");

exit(0);

}

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

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

else if((\*e) == '(') push(\*e,arr,&top);

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

while((x = pop(arr,&top)) != '(') {

printf("%c ",x);

}

}

else {

while(precedence(arr[top]) >= precedence(\*e)) {

printf("%c ",pop(arr,&top));

}

push(\*e,arr,&top);

}

e++;

}

while(top != -1) {

printf("%c ",pop(arr,&top));

}

}

Output

Enter EXP: A+(B/C-(D\*E$F)+G)\*H

A B C / D E F $ \* - G + H \* +

10) Linear Queue using Array

#include<stdio.h>

#include<stdlib.h>

struct Queue

{

int \*arr;

int size;

int front;

int rear;

};

struct Queue create(int size)

{

struct Queue q;

q.size = size;

q.front = 0;

q.rear = 0;

q.arr = (int\*)malloc(sizeof(int)\*size);

if (q.arr == NULL) {

printf("Memory allocation failed\n");

exit(1);

}

return q;

};

void enqueue(struct Queue \*q)

{

int val;

if(q->rear == q->size) printf("\nOVERFLOW\n");

else {

if(q->front == 0 && q->rear == 0) q->front = q->rear = 1;

else (q->rear)++; //normal case

printf("ENTER ELEMENT TO INSERT: ");

scanf("%d",&val);

q->arr[q->rear] = val;

}

}

void dequeue(struct Queue \*q)

{

if((q->front) == 0) printf("\nQUEUE EMPTY!\n");

else if((q->front) == (q->rear)) { //QUEUE contains only one element.

q->front = q->rear = 0;

}

else (q->front)++; //normal case

}

void display(struct Queue \*q)

{

int i;

if((q->front) == 0) printf("\nQUEUE EMPTY!\n");

else{

printf("\nQUEUE ELEMENTS:\n");

for(i=q->front;i<=q->rear;i++) {

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

}

printf("\n");

}

}

int main()

{

struct Queue q;

int ch,val;

printf("ENTER SIZE: ");

scanf("%d",&q.size);

q = create(q.size);

while (1) {

printf("1->INSERT\n");

printf("2->DELETE\n");

printf("3->DISPLAY\n");

printf("4->EXIT\n");

printf("Enter your choice: ");

scanf("%d", &ch);

switch (ch) {

case 1:

enqueue(&q);

break;

case 2: dequeue(&q);

break;

case 3: display(&q);

break;

case 4: free(q.arr);

exit(0);

break;

default:

printf("INVALID CHOICE!\n");

}

}

return 0;

}

Output

ENTER SIZE: 4

1->INSERT

2->DELETE

3->DISPLAY

4->EXIT

Enter your choice: 1

ENTER ELEMENT TO INSERT: 2

1->INSERT

2->DELETE

3->DISPLAY

4->EXIT

Enter your choice: 1

ENTER ELEMENT TO INSERT: 4

1->INSERT

2->DELETE

3->DISPLAY

4->EXIT

Enter your choice: 1

ENTER ELEMENT TO INSERT: 8

1->INSERT

2->DELETE

3->DISPLAY

4->EXIT

Enter your choice: 1

ENTER ELEMENT TO INSERT: 10

1->INSERT

2->DELETE

3->DISPLAY

4->EXIT

Enter your choice: 1

OVERFLOW

1->INSERT

2->DELETE

3->DISPLAY

4->EXIT

Enter your choice: 3

QUEUE ELEMENTS:

2 4 8 10

1->INSERT

2->DELETE

3->DISPLAY

4->EXIT

Enter your choice: 3

QUEUE ELEMENTS:

4 8 10

1->INSERT

2->DELETE

3->DISPLAY

4->EXIT

Enter your choice: 2

1->INSERT

2->DELETE

3->DISPLAY

4->EXIT

Enter your choice: 3

QUEUE ELEMENTS:

8 10

1->INSERT

2->DELETE

3->DISPLAY

4->EXIT

Enter your choice: 2

1->INSERT

2->DELETE

3->DISPLAY

4->EXIT

Enter your choice: 3

QUEUE ELEMENTS:

10

1->INSERT

2->DELETE

3->DISPLAY

4->EXIT

Enter your choice: 2

1->INSERT

2->DELETE

3->DISPLAY

4->EXIT

Enter your choice:

3

QUEUE EMPTY!

1->INSERT

2->DELETE

3->DISPLAY

4->EXIT

Enter your choice: 4

11) Linear Queue using Linked List

#include<stdio.h>

#include<stdlib.h>

typedef struct node

{

int val;

struct node\* next;

}node;

typedef struct queue

{

node\* front;

node\* rear;

}queue;

queue q;

void enqueue(int val)

{

node\* temp;

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

temp->val = val;

temp->next = NULL;

if(q.front == NULL) q.front = q.rear = temp;

else {

q.rear->next = temp;

q.rear = temp;

}

}

void dequeue()

{

if(q.front == NULL) printf("\nQUEUE EMPTY!\n");

else {

node\* temp = q.front;

q.front = q.front->next;

printf("DELETED ELEMENT = %d\n",temp->val);

free(temp);

}

}

void display()

{

node\* curr = q.front;

if(curr == NULL) printf("\nQUEUE EMPTY!\n");

else {

printf("\nQUEUE ELEMENTS:\n");

while(curr!=NULL) {

printf("%d ",curr->val);

curr = curr->next;

}

printf("\n");

}

}

int main()

{

q.front = q.rear = NULL;

int ch,val;

while(1) {

printf("1->INSERT\n");

printf("2->DELETE\n");

printf("3->DISPLAY\n");

printf("4->EXIT\n");

printf("ENTER CHOICE: ");

scanf("%d",&ch);

switch(ch) {

case 1: printf("\nENTER VAL: ");

scanf("%d",&val);

enqueue(val);

break;

case 2: dequeue();

break;

case 3: display();

break;

case 4: exit(0);

default: printf("\nINVALID CHOICE!\n");

}

}

}

Output

1->INSERT

2->DELETE

3->DISPLAY

4->EXIT

ENTER CHOICE: 1

ENTER VAL: 2

1->INSERT

2->DELETE

3->DISPLAY

4->EXIT

ENTER CHOICE: 3

QUEUE ELEMENTS:

2

1->INSERT

2->DELETE

3->DISPLAY

4->EXIT

ENTER CHOICE: 1

ENTER VAL: 5

1->INSERT

2->DELETE

3->DISPLAY

4->EXIT

ENTER CHOICE: 3

QUEUE ELEMENTS:

2 5

1->INSERT

2->DELETE

3->DISPLAY

4->EXIT

ENTER CHOICE: 1

ENTER VAL: 4

1->INSERT

2->DELETE

3->DISPLAY

4->EXIT

ENTER CHOICE: 3

QUEUE ELEMENTS:

5 4

1->INSERT

2->DELETE

3->DISPLAY

4->EXIT

ENTER CHOICE: 2

DELETED ELEMENT = 5

1->INSERT

2->DELETE

3->DISPLAY

4->EXIT

ENTER CHOICE: 3

QUEUE ELEMENTS:

4

1->INSERT

2->DELETE

3->DISPLAY

4->EXIT

ENTER CHOICE: 2

DELETED ELEMENT = 4

1->INSERT

2->DELETE

3->DISPLAY

4->EXIT

ENTER CHOICE: 3

QUEUE EMPTY!

1->INSERT

2->DELETE

3->DISPLAY

4->EXIT

ENTER CHOICE: 4

12) Circular Queue

13) Binary Tree implementation using an Array

#include<stdio.h>

#include<stdlib.h>

typedef struct TreeNode

{

int val;

struct TreeNode \*left;

struct TreeNode \*right;

}TreeNode;

TreeNode\* createNode(int x)

{

TreeNode\* newNode = (TreeNode\*)malloc(sizeof(TreeNode));

newNode->val = x;

newNode->left = NULL;

newNode->right = NULL;

return newNode;

}

TreeNode\* constarr(int arr[],int i,int size)

{

TreeNode\* newNode = NULL;

if(i<size) {

newNode = createNode(arr[i]);

newNode->left = constarr(arr, 2\*i+1, size);

newNode->right = constarr(arr, 2\*i+2,size);

}

return newNode;

}

TreeNode\* insert\_left(TreeNode\* root,int x)

{

root->left = createNode(x);

return root->left;

}

TreeNode\* insert\_right(TreeNode\* root,int x)

{

root->right = createNode(x);

return root->right;

}

void inorder(TreeNode\* root)

{

if(root == NULL) return;

inorder(root->left);

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

inorder(root->right);

}

void preorder(TreeNode\* root)

{

if(root == NULL) return;

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

preorder(root->left);

preorder(root->right);

}

void postorder(TreeNode\* root)

{

if(root == NULL) return;

postorder(root->left);

postorder(root->right);

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

}

int main()

{

int ch,size,i;

printf("Enter number of nodes: ");

scanf("%d",&size);

int arr[size];

printf("enter the elements: ");

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

{

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

}

TreeNode \*root = constarr(arr,0,size);

while(1) {

printf("1->INORDER\n");

printf("2->PREORDER\n");

printf("3->POSTORDER\n");;

printf("4->EXIT\n");

printf("Enter choice: ");

scanf("%d",&ch);

switch(ch) {

case 1: printf("INORDER\n");

inorder(root);

printf("\n");

break;

case 2: printf("PREORDER\n");

preorder(root);

printf("\n");

break;

case 3: printf("POSTORDER\n");

postorder(root);

printf("\n");

break;

case 4: exit(0);

default: printf("\nINVALID CHOICE!\n");

}

}

}

Enter number of nodes: 7

enter the elements: 1

2

3

4

5

6

7

1->INORDER

2->PREORDER

3->POSTORDER

4->EXIT

Enter choice: 1

INORDER

4 2 5 1 6 3 7

1->INORDER

2->PREORDER

3->POSTORDER

4->EXIT

Enter choice: 2

PREORDER

1 2 4 5 3 6 7

1->INORDER

2->PREORDER

3->POSTORDER

4->EXIT

Enter choice: 3

POSTORDER

4 5 2 6 7 3 1

1->INORDER

2->PREORDER

3->POSTORDER

4->EXIT

Enter choice: 4

Output

14) Selection, Bubble, Insertion Sort Menu Driven Program

#include<stdio.h>

#include<stdlib.h>

void swap(int \*a,int \*b)

{

int z = \*a;

\*a = \*b;

\*b = z;

}

void print(int arr[],int n)

{

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

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

}

printf("\n");

}

void selection\_sort(int arr[],int n)

{

for(int i=0;i<n-1;i++) {

int min = i;

for(int j=i+1;j<n;j++) {

if(arr[j]<arr[min]) {

min = j;

}

}

swap(&arr[min],&arr[i]);

}

}

void insertion\_sort(int arr[],int n)

{

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

int key = arr[i];

int j = i - 1;

while(key<arr[j] && j>=0) {

arr[j+1] = arr[j];

--j;

}

arr[j+1] = key;

}

}

void bubble\_sort(int arr[],int n)

{

for(int i=0;i<n-1;i++) {

for(int j=0;j<n-i-1;j++) {

if(arr[j]>arr[j+1]) {

int z = arr[j];

arr[j] = arr[j+1];

arr[j+1] = z;

}

}

}

}

int main()

{

int size;

printf("enter the size of the array: ");

scanf("%d",&size);

int arr[size];

int ch;

while(1) {

printf("0->INSERT ELEMENTS:\n");

printf("1->INSERTION SORT!\n");

printf("2->SELECTION SORT!\n");

printf("3->BUBBLE SORT!\n");

printf("4->EXIT\n");

printf("ENTER CHOICE:\n");

scanf("%d",&ch);

switch(ch) {

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

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

}

break;

case 1: insertion\_sort(arr,size);

print(arr,size);

break;

case 2: selection\_sort(arr,size);

print(arr,size);

break;

case 3: bubble\_sort(arr,size);

print(arr,size);

break;

case 4: exit(0);

default: printf("INVALIDE CHOICE!\n");

}

}

}

Output

enter the size of the array: 5

0->INSERT ELEMENTS:

1->INSERTION SORT!

2->SELECTION SORT!

3->BUBBLE SORT!

4->EXIT

ENTER CHOICE:

0

25

41

78

9

2

0->INSERT ELEMENTS:

1->INSERTION SORT!

2->SELECTION SORT!

3->BUBBLE SORT!

4->EXIT

ENTER CHOICE:

1

2 9 25 41 78

0->INSERT ELEMENTS:

1->INSERTION SORT!

2->SELECTION SORT!

3->BUBBLE SORT!

4->EXIT

0->INSERT ELEMENTS:

1->INSERTION SORT!

2->SELECTION SORT!

3->BUBBLE SORT!

4->EXIT

ENTER CHOICE:

0

2

5

6

3

7

0->INSERT ELEMENTS:

1->INSERTION SORT!

2->SELECTION SORT!

3->BUBBLE SORT!

4->EXIT

ENTER CHOICE:

2

2 3 5 6 7

0->INSERT ELEMENTS:

1->INSERTION SORT!

2->SELECTION SORT!

3->BUBBLE SORT!

4->EXIT

ENTER CHOICE:

0

1

2

3

4

5

0->INSERT ELEMENTS:

1->INSERTION SORT!

2->SELECTION SORT!

3->BUBBLE SORT!

4->EXIT

ENTER CHOICE:

6

INVALIDE CHOICE!

0->INSERT ELEMENTS:

1->INSERTION SORT!

2->SELECTION SORT!

3->BUBBLE SORT!

4->EXIT

4

5

0->INSERT ELEMENTS:

1->INSERTION SORT!

2->SELECTION SORT!

3->BUBBLE SORT!

4->EXIT

ENTER CHOICE:

6

INVALIDE CHOICE!

0->INSERT ELEMENTS:

1->INSERTION SORT!

2->SELECTION SORT!

3->BUBBLE SORT!

4->EXIT

ENTER CHOICE:

3

1 2 3 4 5

0->INSERT ELEMENTS:

1->INSERTION SORT!

2->SELECTION SORT!

3->BUBBLE SORT!

4->EXIT

ENTER CHOICE:

4