Linked list

/\*

Name: Sudheer P

Rollno: 2147234

\*/

#include<stdio.h>

#include<stdlib.h>

//defining structure

struct node

{

int data;

struct node \*next;

};

struct node \*head;

//function declaration

void beginsert ();

void lastinsert ();

void randominsert();

void begin\_delete();

void last\_delete();

void random\_delete();

void display();

void search();

void main ()

{

int choice =0;

printf("\n\t\t\t\t\t\tSINGLY LINKED LIST OPERATIONS\n\n\n");

while(choice != 9)

{

printf("\n1.Insert in begining\n2.Insert at last\n3.Insert at any random location\n4.Delete from Beginning\n5.Delete from last\n6.Delete node after specified location\n7.Search for an element\n8.Show\n9.Exit\n");

printf("\nEnter your choice:\n");

scanf("\n%d",&choice);

switch(choice)

{

case 1:

beginsert();

break;

case 2:

lastinsert();

break;

case 3:

randominsert();

break;

case 4:

begin\_delete();

break;

case 5:

last\_delete();

break;

case 6:

random\_delete();

break;

case 7:

search();

break;

case 8:

display();

break;

case 9:

exit(0);

break;

default:

printf("Please enter valid choice..");

}

}

}

//inserting element in the begining of the linked list

void beginsert()

{

struct node \*ptr;

int item;

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

if(ptr == NULL)

{

printf("\nOVERFLOW");

}

else

{

printf("\nEnter value:\n");

scanf("%d",&item);

ptr->data = item;

ptr->next = head;

head = ptr;

printf("\nNode inserted");

}

}

//inserting element at the end of the linked list

void lastinsert()

{

struct node \*ptr,\*temp;

int item;

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

if(ptr == NULL)

{

printf("\nOVERFLOW");

}

else

{

printf("\nEnter value:\n");

scanf("%d",&item);

ptr->data = item;

if(head == NULL)

{

ptr -> next = NULL;

head = ptr;

printf("\nNode inserted");

}

else

{

temp = head;

while (temp -> next != NULL)

{

temp = temp -> next;

}

temp->next = ptr;

ptr->next = NULL;

printf("\nNode inserted");

}

}

}

//inserting element at random position in the linked list

void randominsert()

{

int i,loc,item;

struct node \*ptr, \*temp;

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

if(ptr == NULL)

{

printf("\nOVERFLOW");

}

else

{

printf("\nEnter element value:");

scanf("%d",&item);

ptr->data = item;

printf("\nEnter the location after which you want to insert :");

scanf("\n%d",&loc);

temp=head;

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

{

temp = temp->next;

if(temp == NULL)

{

printf("\ncan't insert\n");

return;

}

}

ptr ->next = temp ->next;

temp ->next = ptr;

printf("\nNode inserted");

}

}

//deleting element at the begining of the linked list

void begin\_delete()

{

struct node \*ptr;

if(head == NULL)

{

printf("\nList is empty\n");

}

else

{

ptr = head;

head = ptr->next;

free(ptr);

printf("\nNode deleted from the begining ...\n");

}

}

//deleting element at the end of the linked list

void last\_delete()

{

struct node \*ptr,\*ptr1;

if(head == NULL)

{

printf("\nlist is empty");

}

else if(head -> next == NULL)

{

head = NULL;

free(head);

printf("\nOnly node of the list deleted ...\n");

}

else

{

ptr = head;

while(ptr->next != NULL)

{

ptr1 = ptr;

ptr = ptr ->next;

}

ptr1->next = NULL;

free(ptr);

printf("\nDeleted Node from the last ...\n");

}

}

//deleting element at a random position in the linked list

void random\_delete()

{

struct node \*ptr,\*ptr1;

int loc,i;

printf("\n Enter the location of the node after which you want to perform deletion \n");

scanf("%d",&loc);

ptr=head;

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

{

ptr1 = ptr;

ptr = ptr->next;

if(ptr == NULL)

{

printf("\nCan't delete");

return;

}

}

ptr1 ->next = ptr ->next;

free(ptr);

printf("\nDeleted node %d ",loc+1);

}

//search an element in the linked list

void search()

{

struct node \*ptr;

int item,i=0,flag;

ptr = head;

if(ptr == NULL)

{

printf("\nEmpty List\n");

}

else

{

printf("\nEnter item which you want to search?\n");

scanf("%d",&item);

while (ptr!=NULL)

{

if(ptr->data == item)

{

printf("item found at location %d ",i+1);

flag=0;

}

else

{

flag=1;

}

i++;

ptr = ptr -> next;

}

if(flag==1)

{

printf("Item not found\n");

}

}

}

//display element of the linked list

void display()

{

struct node \*ptr;

ptr = head;

if(ptr == NULL)

{

printf("Nothing to print");

}

else

{

printf("\nprinting values . . . . .\n");

while (ptr!=NULL)

{

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

ptr = ptr -> next;

}

}

}

Stack array

#include <stdio.h>

#include<stdlib.h>

#define size 20

int top=-1, my\_arr[size];

int i, arr\_size;

void push();

void pop();

void show();

int main(void) {

int choice;

printf("choose your operation \n 1 push \n 2 pop \n ");

scanf("%d", &choice);

switch(choice){

case 1: push();

break;

case 2: pop();

break;

// case 3: show();

// break;

default: printf("wrong choice");

break;

}

// push();

// pop();

// show();

return 0;

}

void push()

{

printf("Enter the size of the stack: ");

scanf("%d",&arr\_size);

for(i=1; i<arr\_size;i++)

{

printf("insert elements into [%d]= ",i);

scanf("%d", &my\_arr[i]);

top++;

}

}

void pop()

{

if (top ==-1){

printf("Stack is empty: ");

}

else{

printf("deleted elment %d", my\_arr[top]);

top=top-1;

}

}

void show()

{

printf("Element of the stack are \n");

for (i=top; i>-1; i--)

{

printf("Elements in [%d] = %d \n",i, my\_arr[i]);

}

}

Que

//Name:- Sudheer P

//Register number:- 2147234

//Program:- Implementation of enqueue and dequeue

//including the header files:-->

#include<stdio.h>

#include<stdlib.h>

//creating a structure node:-->

struct node

{

int data;

struct node \*next;

};

//creating a pointer pointing to front and rear

struct node \*front = NULL;

struct node \*rear;

//function declaration:-->

void enqueue();

void dequeue();

void display();

//making the main function for the program:-->

void main()

{

int choice;

while (choice!=4)

{

printf("\t\tImplementation of enqueue and dequeue concept");

printf("\nPROGRAM MENU");

printf("\n1.Insert or enqueue an element\n2.Dequeue an element\n3.Display the queue\n4.Exit\n");

printf("\n=============================================================================================");

printf("\nEnter your choice:- ");

scanf("%d",&choice);

//using the switch to switch between opeartions:-->

switch (choice)

{

case 1:

enqueue();

break;

case 2:

dequeue();

break;

case 3:

display();

break;

case 4:

exit(0);

break;

default:

printf("\nSorry it was an invalid request.\nInput a valid entry.");

break;

}

}

}

//defining the function for enqueue:-->

void enqueue()

{

struct node \*ptr;

int item;

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

if(ptr==NULL)

{

printf("\nOverflow");

return;

}

else

{

printf("\nEnter value:\t ");

scanf("%d",&item);

ptr ->data = item;

if (front==NULL)

{

front = ptr;

rear = ptr;

front->next = NULL;

rear->next = NULL;

}

else{

rear->next = ptr;

rear = ptr;

rear->next = NULL;

}

}

}

//defining the function for dequeue:-->

void dequeue()

{

struct node \*ptr;

if (front==NULL)

{

printf("\nUnderflow");

return;

}

else

{

ptr = front;

front = front->next;

free(ptr);

}

}

//defining the display function;-->

void display()

{

struct node \*ptr=NULL;

ptr = front;

if(front==NULL)

{

printf("\nEmpty queue\n");

}

else{

printf("\nPrinting the values......\n");

while(ptr!=NULL)

{

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

ptr = ptr->next;

}

}

}