Date
Page

# program 1: stack

CODE:

```
#include<stdio.h>
char stack[5];
int top=-1;
void push(char ch){
  if(top>=4){
    printf("Stack overflow");
    return;
  }
  stack[++top]=ch;
}
char pop(){
  if(top==-1){
    printf("Stack underflow");
    return " ";
  }
  return stack[top--];
}
char peek(){
  return stack[top];}
char display(){
  for(int i=0;i<=top;i++){
    printf("%c",stack[i]);
  }
}
int main(){
  int x,element;
  printf("1:push 2:peek 3:pop 4:display 5:exit");
  scanf("%d",&x);
  while(x!=5){
    switch(x){
      case1:{
        printf("Enter an element to push:");
```

```
scanf(" %c", &element);
push(element);
break;}
case 2:
    { peek();
    break;}
case 3:{
    printf(" popped element: %c\n", pop());
    break;}
case 4:
    { display();
      break;}
case 5:
    { printf("Exiting... \n");
      break;}
default:
    { printf("Wrong choice! Please try again\n");
      break;}
}
printf("1:push 2:peek 3:pop 4:display 5:exit");
scanf("%d", &x);
}
}
```

**Output:** 1:push 2:peek 3:pop 4:display 5exit 1
Enter an element to push a
1:push 2:peek 3:pop 4:display 5:exit 2
top element: a
1:push 2:peek 3:pop 4:display 5:exit 3
popped element: a
1:push 2:peek 3:pop 4:display 5:exit 1
element to be pushed: b

Date
Page

1:push 2:peek 3:pop 4:display 5:exit 4
ab 1:push 2:peek 3:pop 4:display 5:exit 3
popped element b
1: push 2:peek 3:pop 4:display 5:exit 3
popped element a
1: push 2:peek 3:pop 4:display 5:exit 3
Stack underflow
1:push 2:peek 3:pop 4:display 5:exit 5
Exiting....

# program 2: balanced paranthesis

code:

```
int is_matching_pair(char ch1, char ch2){
    if(ch1=='(' && ch2==')'){
        return 1;
    }else if(ch1=='[' && ch2==']'){
        return 1;
    }else if(ch1=='{' && ch2=='}'){
        return 1;
    }else
        return 0;}

int is_balanced(char* text){
    int i;
    for(i=0;i<strlen(text);i++){
    if(text[i]=='(' || text[i]=='[' || text[i]=='{')
        push(text[i]);
    }else if(text[i]==')' || text[i]==']' ||
                                text[i]=='}'){
        if(top==-1){
            return 0;
        }else if(!is_matching_pair(pop(),text[i])
            return 0;}}}
    if(top==-1){
        return 1;}
    else return 0;
```

output

```
Enter an expression: {[]}
Balanced paranthesis
```