Lab 2 Stacks

|  |  |
| --- | --- |
| Function | Big O |
|  |  |
| int stackSize()  {  return top+1;  } | 2 [addition, return]  O(1) |
| int stackTop()  {  return stack[top];  } | 2 [index, assignment]  O(1) |
| int isEmpty()  {  if(top==-1) return 1;  else return 0;  } | 2 [comparison, assignment]  O(1) |
| int isFull()  {  if(top==stack\_size-1) return 1;  else return 0;  } | 2 [comparison, assignment]  O(1) |
| int push(int x)  {  if(!isFull()) {  top = top + 1;  stack[top] = x;  }  else  return -1;  } | 1[condition]  2[addition, assignment]  1[assignment]  1[return]  O(1) |
| int pop()  { int data;  if(!isEmpty()) {  data = stack[top];  top = top -1;  return data;  }  else  return -1;  } | 1[condition]  1[assignment]  2[subtraction, assignment]  1[return]  1[return]  O(1) |

Exercise 1

|  |  |
| --- | --- |
| int main()  {  int arr[5];  int i;    for(i=0;i<5;i++)  {  arr[i] = i\*2;  }    printf("Old array\n");  for(i=0;i<5;i++)  {  printf("%d ",arr[i]);  }  printf("\n");  stack\_size=10;  stack=(int\*)calloc(sizeof(int), stack\_size);  for(i=0;i<5;i++)  {  push(arr[i]);  }    int newarr[5];    for(i=0;i<5;i++)  newarr[i] = pop();    printf("Reversed array\n");  for(i=0;i<5;i++)  printf("%d ",newarr[i]);    } | 1[Declaration]  1[Declaration]  n+1 (For loop)  n  1[Print]  n+1[For loop]  n  1[Print]  1[Assignment]  1[Declaration]  n+1[For loop]  n  1[Declaration]  n+1(For loop]  n  1[Print]  n+1[For loop]  n  Total: 10n+13  O(n) |
| int main() {  int check = 0;  stack\_size = 100;  stack=(char\*)calloc(sizeof(char), stack\_size);  for(int i=0;i<100;i++)  {  if(br[i]=='[' || br[i]=='{' || br[i]=='(')  {  push(br[i]);  }  if(br[i]==']')  if(pop()!='[')  {  check=1;  break;  }  if(br[i]==')')  if(pop()!='(')  {  check=1;  break;  }  if(br[i]=='}') {  if(pop()!='{')  {  check=1;  break;  }  }    }    if(check)  printf("Incorrect parenthesis");  else  printf("Valid expression");  } | 2[Declaration, Assignment]  1[Assignment]  4[Assignment]  n+1[For loop]  9n[Fetching value at br[i], comparing]  6n[function call + return]  3n[function call + return]  6n[function call + return]  1[Assignment]  3n[function call + return]  6n[function call + return]  1[Assignment]  3n[function call + return]  6n[function call + return]  1[Assignment]  1  O(n) |
| void main()  {  char postfix[25];  char \*ptr;  int n1,n2,n3,num;  printf("Enter the expression: ");  scanf("%s",postfix);  ptr = postfix;  while(\*ptr != '\0')  {  if(isdigit(\*ptr))  {  num = \*ptr - 48;  push(num);  }  else  {  n1 = pop();  n2 = pop();  switch(\*ptr)  {  case '+':  {  n3 = n1 + n2;  break;  }  case '-':  {  n3 = n2 - n1;  break;  }  case '\*':  {  n3 = n1 \* n2;  break;  }  case '/':  {  n3 = n2 / n1;  break;  }  case '^' :  { n3 = pow(n2,n1);  break;  }  }  push(n3);  }  ptr++;  }  printf("\nThe result of expression is %s = %d\n\n", postfix, pop());  } | 1[Declaration]  1[Declaration]  1[Declaration]  1[Declaration]  1[Declaration]  1[Assignment]  n+1[traverses till end of string]  1[print]  O(n) |