**COSE101 COMPUTER PROGRAMMING I**

**2019 FINAL EXAM**

**NAME (이름) : STUDENT ID (학번) : CLASS (분반) :**

1. **(10 points) 다음의 코드를 실행한 뒤 ‘output\_A.txt’와 ‘output\_B.txt’의 내용을 작성하시오.**

**Figure out contents of ‘output\_A.txt’, and ‘output\_B.txt’ after execution of following code.**

|  |  |  |
| --- | --- | --- |
| #include <stdio.h>  #include <string.h>  int main() {  FILE \*fpA = fopen("output\_A.txt", "w");  FILE \*fpB = fopen("output\_B.txt", "w");  char \*strA = ",-/";  char temp[100];  char str[5][60] = { { "summer is coming!" }, // length: 17  { "vacation will let you chill out" }, // length: 31  { "and, have a nice time" }, // length: 21  { "and, stay fit" }, // length: 13  { "and, wish you the best"} }; // length: 22  fprintf(fpA, "%s\n", str[0]);  fprintf(fpB, "%s\n", str[1]);  fclose(fpA);  fclose(fpB);  fpA = fopen("output\_A.txt", "r");  fpB = fopen("output\_B.txt", "w");  \*( strchr(str[2], ' ') ) = '-';  \*(strrchr(str[2], ' ') + 1) = '/';  strtok(str[2], strA);  strcpy(temp, strtok(NULL, strA) );   |  | | --- | | **output\_A.txt:** | | **output\_B.txt:** |   str[1][8] = '\n';  str[1][9] = '\0';  strncat(temp, str[1], strlen(str[1]));    if ( strcmp(str[3], str[4]) )  strcat(temp, str[3]);  else  strcat(temp, str[4]);  fprintf(fpA, "%s", temp);  fprintf(fpB, "%s", temp);  fclose(fpA);  fclose(fpB);  return 0;  } |

1. **(10 points) ‘source.txt’ 파일의 내용이 다음과 같다고 가정하자: 90 87 56 78 -1**

**프로그램이 2번 실행된 후 파일의 내용이 무엇인지 기술하시오.**

**Suppose content of the file ‘source.txt’ is 90 87 56 78 -1**

**Then, what would be the contents of the file after the program is executed twice?**

|  |
| --- |
| #include <stdio.h>  int main() {  FILE \*textfile;  int score;  int s[10];  int n = 0;  int i;  textfile = fopen("source.txt", "r");  if (textfile == NULL) {  printf("Can't open scores.txt\n");  exit(1);  }  for (;;) {  fscanf(textfile, "%i", &score);  if (score == -1) break;  s[n] = score;  n++;  }  fclose(textfile);  s[n] = 99;    textfile = fopen("source.txt", "w");  for (i = n; i > 0; i-=2) {  fprintf(textfile, "%i ", s[i]);  }  fprintf(textfile, "-1\n");  fclose(textfile);  } |

|  |
| --- |
| **Answer:** |

1. **(15 points) 영문자로만 이루어져 있고 반복된 문자가 많이 나오는 문자열의 경우 문자의 반복횟수를 표시하여 문자열의 길이를 줄일 수 있다. 예를 들어 다음과 같이 변환할 수 있다.**

**We can compress strings composed of alphabet characters as follows.**

**• "zzzzzzzz"→ "8z"**

**• "bbbbcceeeee"→ "4b2c5e"**

**한 문자의 반복 횟수는 최대 9 회이고, 문자가 반복되지 않는 경우는 반복횟수를 표시하지 않는다. 또 이 알고리즘은 대문자와 소문자를 구분하고, 공백 문자열은 그대로 둔다.**

**We assume that the maximum number of repetitions of the same alphabet is 9. If different characters are consecutive, it is not compressed. Empty string is compressed as an empty string. The program is case-sensitive.**

**• "abcde"→ "abcde"**

**• "aaaaaAAAA"→ prints "5a4A"**

**• ""→ ""**

|  |
| --- |
| #define BUFSIZE 100  #include <stdio.h>  void strcompress(char\* input, char\* output);  void strdecompress(char\* input, char\* output);  int main() {  char str[] = "BBBBBBBBbbbbbbaaacdddeeeeee";  char comp[BUFSIZE];  char decomp[BUFSIZE];  strcompress(str, comp);  printf("Compress: %s\n", comp);  // Compress: 8B4b2b3ac3d6e  strdecompress(comp, decomp);  printf("Decompress: %s\n", decomp);  // Decompress: BBBBBBBBbbbb bbaaacdddeeeeee  } |

**(a) 주어진 문자열을 압축하는 함수 strcompress 를 작성하시오.**

**Write ‘strcompress’ function that is supposed to compress given strings.**

void strcompress(char\* input, char\* output) {

|  |
| --- |
|  |

}

**(b) 압축된 문자열을 복원하는 함수 strdecompress 를 작성하시오.**

**Write the ‘strdecompress’ function that is supposed to decompress given strings.**

void strdecompress(char\* input, char\* output) {

|  |
| --- |
|  |

}

1. **(15 points) 뒤집어서 더하기 연산은 어떤 양의 정수가 주어졌을 때 그 수를 뒤집어서 더한 값을 출력하는 것이다. 함수 flipadd() 는 이 연산을 수행하며 다음과 같은 결과를 출력한다.**

**The function flipadd() adds a given positive integer and its flipped value as shown below:**

• flipadd(123) = 123 + 321 = 444

• flipadd(195) = 195 + 591 = 786

**이 함수를 몇 차례 반복 적용하면 앞으로 읽으나 뒤로 읽으나 같은 값이 되는 회문수 (palindrome number) 가 된다. 예를 들어 195 를 시작으로 flipadd() 를 몇 차례 적용하면 회문수인 9339 가 출력된다.**

**With repeated applications, a number often eventually becomes a palindrome number. For example, applying flipadd() to the number 195 becomes 9339, which is a palindrome number.**

**There are several more palindrome numbers as follows:**

• flipadd(195) = 195 + 591 = 786

• flipadd(786) = 786 + 687 = 1473

• flipadd(1473) = 1473 + 3741 = 5214

• flipadd(5214) = 5214 + 4125 = 9339

**대부분의 정수는 이 단계를 몇 단계만 반복하면 회문수가 된다. 그렇지만 어떤 수는 1,000 번 정도 반복해도 회문수가 되지 않는다. 숫자를 입력으로 받아 입력 값, 그 숫자가 회문수가 되는데**

**소요되는 반복횟수, 그리고 마지막으로 계산된 회문수를 출력하는 프로그램을 작성하시오.**

**단, 함수 flipadd()는 recursion으로 구현 하시오.**

**Most integers become a palindrome number after applying the function several times. On the contrary, there are numbers which do not become a palindrome number with even more than 1,000 iterations. Write a program which outputs the initial value (e.g., input), number of iterations necessary for it to become a palindrome number, and the palindrome number.**

**For flipadd() function, you must use recursion to implement it.**

**Sample Input**

195

196

**Output**

195 4 9339

Out of Range : 196 1001 -1403241137

|  |
| --- |
| #include <stdio.h>  int flip(int n);  int is\_palin(int n);  int flipadd(int n, int\* count);  void main() {  int num, result;  int count = 0;  scanf\_s("%d", &num);    result = flipadd(num, &count);  printf("%d %d %d\n", num, count, result);    } |

int flip(int n) {

|  |
| --- |
|  |

}

int is\_palin(int n) {

|  |
| --- |
|  |

}

int flipadd(int n, int\* count) {

|  |
| --- |
|  |

}

1. **(20 points) 10명의 사람이 원을 이루면서 앉아있다고 가정하자. 양의 정수 K(≤ 10)에 대하여, K번째 사람을 제거한다. 원 안의 모든 사람이 제거될때까지 이 작업을 반복할 때, 이 과정에서 사람들이 제거되는 순서를 K-요세푸스 순열이라고 한다.**

**Suppose there are 10 people sitting around. Given a positive integer K(≤ 10), every K-th person is removed from the circle at a time. If we do this procedure until everyone is removed from the circle, the order of the removal is called K-Josephus permutation.**

**각 사람의 번호와 이름 쌍으로 이루어진 structure 들은 다음과 같다.**

**A set of the number and name of each person is stored as a structure as follows:  
1. Alice 2. Bob 3. Charlie 4. Dave 5. Eve 6. Faythe 7. Grace 8. Heidi 9. Ivan 10. Justin**

**이 경우, 3-요세푸스 순열은 <3-Charlie, 6-Faythe, 9-Ivan, 2-Bob, 7-Grace, 1-Alice, 8-Heidi, 5-Eve, 10-Justin, 4-Dave> 이다.**

**As an example, 3-Josephus permutation is <3-Charlie, 6-Faythe, 9-Ivan, 2-Bob, 7-Grace, 1-Alice, 8-Heidi, 5-Eve, 10-Justin, 4-Dave>.**

**아래에 같이 주어진 프로그램에 대해, (a)와 (b)에 답하시오.**

**Given the following program, solve the problems (a) and (b).**

|  |
| --- |
| #include <stdio.h>  #include <stdlib.h>  #include <string.h>  typedef struct {  int num;  char name[20];  }josephus;  struct listNode {  josephus value;  struct listNode \*link;  };  typedef struct listNode\* ListNodePtr;  void printList(ListNodePtr \*sPtr) {  ListNodePtr currentP = \*sPtr;  while (currentP != NULL) {  printf("%d : %s ", currentP->value.num, currentP->value.name);  currentP = currentP->link;  }  }  int main() {  josephus player[10] = {  { 1,"Alice" },  { 2,"Bob" },  { 3,"Charlie" },  { 4,"Dave" },  { 5,"Eve" },  { 6,"Faythe" },  { 7,"Grace" },  { 8,"Heidi" },  { 9,"Ivan" },  { 10,"Justin" }  };  josephus result[10] = { 0 };  int K;  ListNodePtr startPtr = NULL;  printf("Input Number K : ");  scanf("%d", &K);  josephus\_Permute(player, K, result);  for (int i = 0; i < 10; i++) {  josephus\_Insert(&startPtr, result[i]);  }  printList(&startPtr);  return 0;  } |

1. **주어진 K에 대하여, 배열 ‘player’의 K-요세푸스 순열을 배열 ‘result’ 에 저장하는 함수 josephus\_Permute 를 작성하라. 예를 들어 3-요세푸스 순열의 경우, result[0]에 ‘3’과 ‘Charlie’를 저장하고, result[1]에는 ‘6’과 ‘Faythe’를 저장한다. 각 배열의 크기는 10으로 한다.**

**Write the josephus\_Purmute function that stores K-Josephus permutation of the array ‘player’ into the array ‘result’. In case of 3-Josephus permutation, for example, result[0] stores ‘3’ and ‘Charlie’, and result[1] stores ‘6’ and ‘Faythe’.** **The size of each array is 10.**

**K = 3일 때, 배열 ‘result’에 저장된 값은 다음과 같다.  
If K = 3, stored values of array ‘result’ is as a follow.  
 result[10]** = {  
 { 3,"Charlie" },  
 { 6,"Faythe" },  
 { 9,"Ivan" },  
 { 2,"Bob" },  
 { 7,"Grace" },  
 { 1,"Alice" },  
 { 8,"Heidi" },  
 { 5,"Eve" },  
 { 10,"Justin" },  
 { 4,"Dave" }  
 }

void josephus\_Permute(josephus Player[], int K, josephus result[]) {

|  |
| --- |
|  |

}

1. **정렬된 result 배열의 값을 순서대로 링크드리스트에 추가하여 K-요세푸스 리스트를 구성하려 한다. 링크드 리스트에 K-요세푸스 순열 순서대로 노드 하나를 추가하는 함수 josephus\_Insert 를 작성하여라.**

**We aim to construct K-Josephus list by adding the values ​​of the sorted ‘result’ array to a linked list. Write the josephu\_Insert() function that inserts the element of array ‘result’ into a linked list.**

void josephus\_Insert(ListNodePtr \*sPtr, josephus result\_i) {

|  |
| --- |
|  |

}

1. **(30 points) 다음과 같이 링크드리스트와 main 함수가 주어져 있다. 메인 함수와 리스트 노드를 참고하여 (a)와 (b)의 기능을 하는 함수를 작성하시오.**

**Given the following linked list and main(), solve the problems (a) and (b).**

**리스트 노드의 구성은 다음과 같다:** 1->2->3->4->5->6->7->8->NULL

**Linked list is constructed as a follow:** 1->2->3->4->5->6->7->8->NULL

|  |
| --- |
| #include<stdio.h>  #include<stdlib.h>  struct listNode {  int data;  struct listNode\* link;  };  typedef struct listNode\* ListNodePtr;  void addtohead(ListNodePtr \*sPtr, int new\_data){  // insert a node into the linked list  }  void printList(ListNodePtr \*sPtr)  {  ListNodePtr currentP = \*sPtr;  while (currentP != NULL)  {  printf("%d ", currentP->data);  currentP = currentP->link;  }  }  int main(void)  {  ListNodePtr startPtr = NULL;  for (int i = 8; i > 0; i--)  addtohead(&startPtr, i);  printf("Given linked list \n");  printList(startPtr);  startPtr = flipList(&startPtr, 3);  printf("\nReversed Linked list \n");  printList(startPtr);  rotateList(&startPtr, 4);  printf("\nRotated Linked list \n");  printList(startPtr);  return 0;  } |

1. **리스트가 주어졌을 때, N(>0)단위로 나누어 Flip하는 함수 fliplist()를 Recursion을 사용하여 작성하시오. 남은 리스트가 N보다 작은 경우, 남은 리스트만 Flip한다. 단, addtohead() 함수는 사용할 수 없다.**

**Given a linked list and N(>0), write a recursive function fliplist() that flips a block of N numbers reversely. If the remaining number is less than N, just flip the remaining ones. You should use addtohead() function to implement it.**

**For example,**

**N=1: 1->2->3->4->5->6->7->8->NULL 🡺 1->2->3->4->5->6->7->8->NULL**

**N=4: 1->2->3->4->5->6->7->8->NULL 🡺 4->3->2->1->8->7->6->5->NULL**

**N=3: 1->2->3->4->5->6->7->8->NULL** 🡺 **3->2->1->6->5->4->8->7->NULL**

**N=10: 1->2->3->4->5->6->7->8->NULL 🡺 8->7->6->5->4->3->2->1->NULL**

ListNodePtr flipList(ListNodePtr \*sPtr, int N){

|  |
| --- |
|  |

}

1. **리스트가 주어졌을 때, N(>0)번째 까지의 리스트 구성 요소를 나머지 리스트의 뒤로 이동시키는 rotateList() 함수를 작성하시오. 단, addtohead() 함수는 사용할 수 없다.**

**Given a linked list and a positive integer N(>0), write a function rotateLIst() that moves the first N numbers into the end of the list. You should not use addtohead() function to implement it.**

**For example,  
N=1: 1->2->3->4->5->6->7->8->NULL 🡺 2->3->4->5->6->7->8->1->NULL  
N=4: 1->2->3->4->5->6->7->8->NULL** 🡺 **5->6->7->8->1->2->3->4->NULL  
N=6: 1->2->3->4->5->6->7->8->NULL** 🡺 **7->8->1->2->3->4->5->6->NULL  
N=15: 1->2->3->4->5->6->7->8->NULL** 🡺 **1->2->3->4->5->6->7->8->NULL**

void rotateList(ListNodePtr \*sPtr, int N){

|  |
| --- |
|  |

}